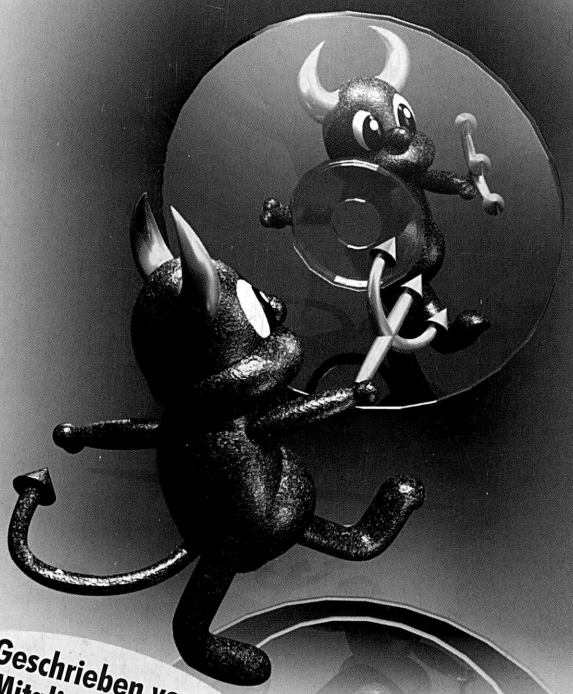
Computer & Literatur C&L

Herausgegeben von der Redaktion freeX

NetBSD 1.6

Installieren • Konfigurieren • Administrieren



Geschrieben von Mitgliedern der NetBSD Foundation und erfahrenen Anwendern

Deutsche Originalausgabe

NetBSD 1.6

NetBSD 1.6

Herausgegeben von der Redaktion

freeX

C&L

Computer & Literatur Verlag GmbH

Die Deutsche Bibliothek – CIP-Einheitsaufnahme
Redaktion freeX (Hrsg.):
NetBSD 1.6
[Medienkombination]
ISBN 3-936546-00-2

Alle Rechte vorbehalten. Ohne ausdrückliche, schriftliche Genehmigung des Herausgebers ist es nicht gestattet, das Buch oder Teile daraus in irgendeiner Form durch Fotokopie, Mikrofilm oder ein anderes Verfahren zu vervielfältigen oder zu verbreiten. Dasselbe gilt für das Recht der öffentlichen Wiedergabe.

Der Verlag macht darauf aufmerksam, daß die genannten Firmen- und Markenzeichen sowie Produktbezeichnungen in der Regel marken-, patent-, oder warenzeichenrechtlichem Schutz unterliegen.

Die Herausgeber übernehmen keine Gewähr für die Funktionsfähigkeit beschriebener Verfahren, Programme oder Schaltungen.

2005 04 03
6 5 4 3 2 1

© 2003 by C&L Computer und Literaturverlag
Zavelsteiner Straße 20, 71034 Böblingen
E-Mail: info@cul.de
WWW: http://www.cul.de

Coverdesign: Hawa & Nöh, Kassel
Satz: C&L Computer und Literaturverlag, Böblingen
Druck: PUT i RB DROGOWIEC
Printed in Poland

Dieses Buch wurde auf chlorfrei gebleichtem Papier gedruckt

ISBN 3-936546-00-2

Inhalt

Vorwort des Herausgebers *13*

1 Das ist NetBSD *15*
von Thorsten Lindloff

2 NetBSD installieren *23*
von Thorsten Lindloff, Jörg Braun, Ignatios Souvatzis, Felix J. Ogris und Kevin Read

2.1 Die Installation auf dem Intel-PC 23
2.1.1 Die Installation von CD-ROM 23
2.1.2 Die Installation über das Netzwerk 44
2.1.3 Die Installation per FTP 45
2.1.4 Die Grundkonfiguration 45
2.1.5 Das Booten des installierten NetBSD-Systems 48
2.2 Die Installation auf dem Mac68k 50
2.3 Die Installation auf dem Amiga 70
2.3.1 Systemvoraussetzungen 70
2.3.2 Vorbereitung der Installation 71
2.3.3 Die eigentliche Installation 78
2.3.4 Nachbereitung 82
2.3.5 Upgrade statt Installation 87
2.4 Die Installation auf der Sparc 87
2.5 Die Installation auf der DECStation 100
2.5.1 Installation über das Netzwerk 104
2.5.2 Installation über eine Festplatte oder ein CD-ROM 107
2.5.3 So geht es weiter 107

3 NetBSD konfigurieren *111*
von Thorsten Lindloff, Jörg Braun, Ignatios Souvatzis und Dr. Lex Wennmacher

3.1 Grundkonfiguration des Systems 111
3.1.1 Der Aufbau des NetBSD-Verzeichnisbaums 111
3.1.2 vi, das wichtigste Programm 111
3.1.3 Die Tastatur an lokale Gegebenheiten anpassen 113
3.1.4 Das Passwort 115
3.1.5 Wechseln der Shell 115
3.1.6 Weitere Benutzer anlegen 118
3.1.7 root-Passwort vergessen 122
3.1.8 Das rc-Startsystem in NetBSD 123
3.1.9 Das Netzwerk und seine Basiskonfiguration 127
3.1.10 Software nachinstallieren 131

3.2 **Das Programm sushi** ... 132
3.2.1 Softwareverwaltung ... 134
3.2.2 Systemverwaltung ... 137
3.2.3 Sicherheit und Benutzer ... 140
3.2.4 Prozesse und Daemonen in sushi ... 141
3.2.5 Netzwerk-Konfiguration ... 142
3.3 **Datenträgerverwaltung auf dem i386** ... 143
3.3.1 Festplattenaufteilung mit fdisk ... 147
3.3.2 Das BSD-Disklabel ... 151
3.3.3 Das Spezialverzeichnis »/dev« ... 162
3.3.4 Datenträgeraustausch mit dem Macintosh ... 164
3.4 **Der NetBSD/i386-Bootselektor** ... 166
3.5 **Absichern des Systems mit File Flags und Securelevel** ... 169

4 Der Kernel ... ***175***
von Martin Laubach und Jörg Braun

4.1 **Der Kernel-Baukasten** ... 175
4.1.1 Vorbereitungen ... 176
4.1.2 Konfigurationsdateien ... 176
4.1.3 Kompilation ... 178
4.1.4 Im Notfall ... 180
4.1.5 Struktur der Hardware ... 180
4.2 **Die Konfigurationsdatei** ... 183
4.3 **Die Kernel-Konfiguration für ISDN** ... 235
4.3.1 Die Grundkonfiguration ... 235
4.3.2 ISDN-Karten ... 236
4.4 **Das Programm sysctl** ... 241

5 Die Package-Sammlung ... ***253***
von Markus Illenseer und Jörg Braun

5.1 **Installation und Wartung auf der Kommandozeile** ... 256
5.1.1 Schnellinstallation über Binärpakete ... 257
5.1.2 Das Package-Infosystem ... 259
5.1.3 Packages löschen ... 259
5.1.4 Die Paket-Hierarchie ... 260
5.2 **Installation mittels Package-Sourcetree** ... 261
5.2.1 Installation von Packages ... 263
5.2.2 Aufräumarbeiten ... 264
5.2.3 Installations-Optionen ... 265
5.2.4 Eigene Binär-Packages erzeugen ... 265
5.2.5 Ressourcen ... 266
5.3 **NetBSD-Pakete und KDE** ... 267
5.4 **Aktualisieren des Package Source** ... 271

5.5 Überblick verschaffen ... 273
5.5.1 Packages und Firewall ... 276
5.5.2 Mailingliste ... 276
5.6 Lizenzbedingungen ... 276
5.7 Fehlermeldungen ... 278
5.8 Das Bauen eigener Package-Source-Verzeichnisse ... 279
5.9 Das Bauen eigener Binärpackages ... 286
5.10 Der Grundaufbau des Package-Sourcetrees ... 288

6 Das Betriebssystem aktualisieren ... 293
von Dr. Lex Wennmacher

6.1 Das Standardverfahren ... 293
6.2 Das Alternativverfahren ... 296

7 Das Netzwerk ... 299
von Markus Illenseer, Ignatios Souvatzis, S.P. Zeidler, Dr. Lex Wennmacher, Thorsten Lindloff und Thorsten Stritzinger

7.1 Die Grundlagen ... 299
7.1.1 Die Konfigurationsdateien ... 300
7.1.2 Einbindung eines Rechners in ein lokales Netz ... 303
7.1.3 Die Wartung ... 303
7.1.4 Probleme mit dem Netzwerk ... 304
7.1.5 inetd, der Internet Super Server ... 304
7.1.6 Einen Dienst hinzufügen ... 307
7.2 Die Grundlagen von TCP/IP ... 308
7.2.1 TCP/IP v4 ... 308
7.2.2 Einführung in IPv6 ... 312
7.3 DNS und andere Netz-Konfigurationen ... 319
7.3.1 Netzwerkkonfiguration als DHCP-Client ... 319
7.3.2 Statische Netzwerkadressen ... 319
7.4 NetBSD als Firewall und Router ... 331
7.4.1 NetBSD-Router mit zwei Netzwerkkarten ... 331
7.4.2 Installation als Application-Gateway ... 333
7.4.3 IP-NAT installieren ... 334
7.5 Sicherheit mit ssh und ssl ... 337
7.5.1 ssh – Die Secure Shell ... 337
7.5.2 Apache-Server mit ssl installieren ... 339
7.6 Dienste von TCP/IP ... 348
7.6.1 Installation des ftp-Servers ... 348
7.6.2 Konfigurieren eines Anonymous ftp-Servers ... 353
7.6.3 Mit NTP die Zeit im Griff ... 359
7.6.4 Der ssh-Daemon ... 375
7.6.5 Yellow Pages ... 377
7.6.6 Sendmail ... 385

8 Wählverbindungen *401*
von Karl Uwe Lockhoff

8.1 **Mit ppp ins Internet** 402
8.1.1 Kernel-Anforderungen 402
8.1.2 Wichtige Konfigurationsdateien 402
8.2 **Ein Netzwerk mit dem Internet verbinden** 408
8.3 **Mit ISDN ins Internet** 411
8.3.1 Die Kernelkonfiguration für ISDN 411
8.3.2 Konfiguration des Zugangs 412
8.3.3 ISDN bei Bedarf 414
8.4 **ppp über das Ethernet – ADSL-Verbindungen** 415
8.5 **Kernelkonfiguration** 415
8.5.1 Permanente Verbindungen (Flatrate) 417
8.5.2 Einwahl bei Bedarf 418
8.6 **WaveLAN – drahtlos verbinden** 420
8.6.1 Kernelkonfiguration 420
8.6.2 Setup fürs Netzwerk 420
8.6.3 Erstellen der Konfigurationsdateien 422
8.7 **VPN (Virtual Private Network) mit ppp über SSH** 422

9 Datenaustausch im Netz *427*
von Dr. Lex Wennmacher, Thorsten Lindloff, Jörg Braun und Michael Steil

9.1 **Datenaustausch via NFS** 427
9.1.1 Vorbereitende Arbeiten 428
9.1.2 NFS-Server einrichten 430
9.1.3 NFS-Client einrichten 432
9.1.4 Selten benötigt 435
9.1.5 Probleme finden und beseitigen 437
9.1.6 Leistungssteigerung von NFS 438
9.2 **Der Samba-Server** 441
9.2.1 Das Domänensystem von Windows 444
9.2.2 Einschränkungen von Samba 446
9.2.3 Die Namensauflösung in Windows-Netzen 446
9.2.4 Die Freigabe von Netzwerklaufwerken 449
9.2.5 Netzwerkdrucker 449
9.2.6 Die Installation 449
9.2.7 Samba auf die Schnelle 452
9.2.8 Samba verstehen 457
9.2.9 Samba und seine Benutzer 461
9.2.10 Die Security-Einstellungen 463
9.2.11 Drucken mit Samba 464
9.2.12 Das Browsing in einer Arbeitsgruppe 465
9.2.13 Die zentrale Benutzerverwaltung 466
9.2.14 Der Anmeldeprozeß 467
9.2.15 Einrichten eines Domain Controllers 468
9.2.16 Die Konfiguration von Samba mit SWAT 471

9.2.17 Die grafische Konfiguration ... 472
9.3 Samba-Clients unter NetBSD ... 473
9.4 MacOS 8/9 im Unix-Netz mit Netatalk ... 477
9.6 AFS ... 490

10 Die Shell ... 491
von Alexander Mayer

10.1 Überblick über die Shells ... 491
10.2 Grundlagen ... 493
10.3 Dateinamenexpandierung ... 503
10.4 Ein- und Ausgabeverarbeitung ... 508
10.5 Aliase ... 513
10.6 Die History ... 517
10.7 Eingabe editieren ... 523
10.8 Die Jobverwaltung ... 533
10.9 Standardvariablen ... 536
10.10 Shelloptionen ... 543
10.10.1 Shelloptionen der Bourne-Shell ... 543
10.10.2 Shelloptionen der Korn-Shell ... 544
10.10.3 Shelloptionen der Bash ... 545
10.10.4 Shelloptionen der C-Shell ... 550
10.11 Konfigurationsdateien ... 550
10.11.1 Einführung ... 550
10.11.2 Die Konfigurationsdateien der Bourne-Shell ... 551
10.11.3 Die Konfigurationsdateien der C-Shell ... 551
10.11.4 Die Konfigurationsdateien der Korn-Shell ... 552
10.11.5 Die Konfigurationsdateien der Bash ... 553
10.12 Ausblick ... 554

11 Befehle der Kommandozeile ... 555
von Norbert Braun

11.1 Allgemeine Befehle ... 555
11.2 Leben im Verzeichnisbaum ... 557
11.3 Wer darf was mit wem? ... 568
11.4 Bearbeitung von Textdateien ... 571
11.5 Manipulation von Dateien ... 576
11.6 Leben im Netz ... 580
11.7 Sonstige Kommandos ... 582
11.8 Datensicherung ... 585
11.9 Prozeßverwaltung ... 591
11.10 Für den Administrator ... 596
11.11 Interessante Tools ... 603
11.12 Nützliche Zusatzpakete ... 606

12 X Window *609*
von Wolfgang Soltendick, Martin Schulze und Jörg Braun

12.1 Installation 609
12.2 Der Verzeichnisaufbau 610
12.3 Konfiguration 611
12.3.1 xf86config 611
12.3.2 xf86cfg 614
12.3.3 Manuelle Bearbeitung 624
12.3.4 Beispiel 627
12.4 Start des Default-Systems 630
12.5 Die Konfigurationsdateien von X 632
12.6 Nachträgliche Konfiguration 633
12.6.1 xf86cfg 633
12.6.2 xvidtune 638
12.7 Von Version 4 nicht unterstützte Karten 643
12.7.1 Framebuffer-Treiber 643
12.7.2 Server der Version 3.3.6 643
12.7.3 Erster Start 651
12.8 Die mitgelieferten Fenstermanager 652
12.8.1 Afterstep 652
12.8.2 blackbox 653
12.8.3 9wm 653
12.8.4 Enlightenment 654
12.8.5 sawfish 654
12.8.6 WindowMaker 655
12.9 Weitere Fenstermanager 656
12.10 KDE 3 – K Desktop Environment 658
12.10.1 Einrichtung 658
12.10.2 Konfiguration und Start 661
12.10.3 Minimalsystem 663
12.11 Gnome 664
12.11.1 Einrichtung 664
12.11.2 Konfiguration und Start 665
12.11.3 Minimalsystem 668
12.12 X für Puristen: Der twm 669
12.13 X und die Tastatur 679
12.14 Display-Manager 682
12.14.1 xdm 682
12.14.2 gdm 684
12.14.3 kdm 692
12.15 X im Netz 694
12.15.1 Ausgabe umlenken 695
12.15.2 Umleitung mit »xhost« 696
12.15.3 Umleitung mit »auth« 697
12.15.4 Umleitung mit »ssh« 699
12.16 Alternative automatische X-Konfiguration 699

13 Drucken mit NetBSD 703
von Thorsten Lindloff und Dr. Bernd Kallenbach

13.1 Drucken mit dem lpd 703
13.1.1 Ghostscript 705
13.1.2 Der Druckvorgang 705
13.1.3 Magicfilter installieren 709
13.1.4 Drucken im Unix-Netz 710
13.2 Das Common Unix Printing System (CUPS) 714
13.2.1 Kernel-lp-Device (Druckerport) 716
13.2.2 Unix-Printing-System 716
13.2.3 Drucker-Spooling 717
13.2.4 Filter 718
13.2.5 Ghostscript 718
13.2.6 Installation und Administration von CUPS 721
13.2.7 Administration 722
13.2.8 Drucker-Klassen in CUPS 724
13.2.9 Druckerinstanzen in CUPS 724
13.2.10 Netzwerkdruck mit CUPS 725
13.2.11 Das Web-Interface 725
13.2.12 Administrative Grafiktools 726

14 Das Init-Konzept von NetBSD 729
von Wolfgang Soltendick

15 Administration mit Webmin 739
von Georg Basse

15.1 Historie 739
15.2 Aufbau und Arbeitsweise 740
15.3 Installation 741
15.4 Starten und Stoppen von Webmin 743
15.5 Serverindex 744
15.6 Java-Applets 747
15.7 Überwachung 749
15.8 Die Benutzerverwaltung 749
15.9 cron 750
15.10 Die Softwareverwaltung 751
15.11 Start und Stop von Programmen 752
15.12 Die Druckerverwaltung 753
15.13 Internetdienste 756
15.13.1 Namensauflösung konfigurieren 757
15.13.2 Sicherheitsaspekte 758
15.13.3 Sendmail konfigurieren 759
15.13.4 Virtuelle Benutzer 760
15.13.5 Das Apache-Modul 761
15.13.6 Zugangskontrolle 763

15.13.7 Ablage der Webdaten 764
15.13.8 WU-FTPD administrieren 764

16 Der NetBSD-Sourcetree 767
von Jörg Braun

16.1 Das Kompilieren von Verzeichnissen 769
16.2 build.sh und Cross Compile 775

Anhang A: Spezielle Anwendungsbereiche 779
von Ignatios Souvatzis und Jörg Braun

A.1 Schreibgeschützter Rootbereich 779
A.2 Die CD-ROM zum Buch: NetBSD Live! 784
A.2.1 Der Kernel und das Booten 784
A.2.2 Das CD-Dateisystem 788
A.2.3 X von CD 798
A.2.4 Anwendungen 800
A.2.5 Mastern und brennen von CDs 802

Anhang B: Systembefehle von NetBSD 815

Stichwortverzeichnis 840

Vorwort des Herausgebers

NetBSD, der Name »Net« sagt es aus, ist ein Betriebssystem des Internet-Zeitalters. Personen in aller Welt entwickeln es weiter und setzen es ein. Und ein NetBSD-Server soll der am längsten laufende Internet-Server der Welt sein[1]. NetBSD ist dabei mehr als einfach ein weiteres Betriebssystem, das man auf dem PC installieren und einsetzen kann: es steht auf über vierzig verschiedenen Plattformen zur Verfügung. Damit ist NetBSD das am meisten portierte Betriebssystem überhaupt. Dies beweist, wie sauber NetBSD aufgebaut sein muß. Nur durch eine klare Trennung der Hardware-Schicht und der darauf aufbauenden Routinen kann ein Betriebssystem auf völlig unterschiedlicher Hardware und dabei auf einem halben Dutzend CPUs von Intel über Motorola 68000 und PowerPC über MIPS und SPARC bis zum exotischen RISC-Prozessor laufen.

Wenn Sie NetBSD einsetzen wollen, sind Sie also nicht auf den »normalen« Durchschnitts-PC beschränkt. In diesem Buch lesen Sie, wie Sie es auch auf einem alten Macintosh, auf einer DECstation, Sparc-Station oder auf einem Amiga installieren und betreiben.

Ich selbst verfolge die Entwicklung von NetBSD schon einige Jahre. Obwohl es sich damals Version 1.2 und heute Version 1.6 nennt und der »Versionssprung« minimal erscheint, liegen dazwischen Welten. Die Versionsnummer drückt nicht den wirklichen Stand im Vergleich zu anderen Open-Source-Betriebssystemen aus. Wenn man bei der Entwicklergruppe mehr Wert auf engagiertes Marketing legen würde, müßte heute mindestens eine 5 oder 6 vor dem Punkt stehen. Aber im Gegensatz zu Linux-Distributoren steht hinter NetBSD kein finanzielles Interesse, vielmehr der Wunsch, ein stabiles, portables Betriebssystem zu entwickeln.

Was an NetBSD immer wieder fasziniert, ist die Qualität des Kernels. Bei keinem anderen Betriebssystem ist die Hardware-Erkennung so intelligent und ausgefeilt. Daß es keine bunten Wartungsprogramme gibt, bedeutet einen geringfügig höheren Arbeitsaufwand im beim ersten Einrichten im Vergleich zu anderen Betriebssystemen. Aber anschließend weiß man wenigstens, was passiert. Hier wird nichts hinter Menüs versteckt oder dem Anwender und Administrator vorenthalten und läuft dann »einfach so« ab.

NetBSD ist offen und transparent wie kein anderes Betriebssystem. Es macht Spaß, damit zu arbeiten. Wenn Sie sich nicht trauen, es gleich zu installieren, probieren Sie die CD-ROM, die wir für dieses Buch gemacht haben. Sie bootet ein vollwertiges NetBSD 1.6 auf PCs ab Pentium-Prozessor direkt und ohne Installation. Wir haben dem Buch absichtlich keine Installationsdistribution des Betriebssystems mitgegeben. Sie können sich das Grundsystem für die gewünschte Plattform direkt als ISO-Image vom NetBSD-Server laden oder eine der preiswerten Teil- oder Komplettdistributionen bei beliebigen Händlern vor Ort oder im Internet bestellen. Sie sind dann auf jeden Fall auf dem momentan aktuellen Stand des Betriebssystems.

---

[1] laut *http://www.uptimes.nu/*

Die Autoren dieses Buchs sind teilweise Mitglieder der NetBSD-Gruppe selbst und ihre Namen und Adressen tauchen auch in der Mitgliederliste auf dem NetBSD-Webserver wieder auf. Die anderen Autoren sind langjährige erfahrene BSD-Anwender. Mein herzlicher Dank dafür, daß sie dieses Buch, auf das die BSD-Gemeinde sehnsüchtig gewartet hat, geschrieben haben, gilt ihnen allen: *Georg Basse, Jörg Braun, Marcus Illenseer, Dr. Bernd Kallenbach, Martin Laubach, Thorsten Lindloff, Karl-Uwe Lockhoff, Alexander Mayer, Felix J. Ogris, Kevin Read, Wolfgang Soltendick, Ignatios Souvatzis, Thorsten Stritzinger, Dr. Lex Wennmacher und S.P. Zeidler.*

Rosa Riebl

Herausgeberin

1 Das ist NetBSD

von Thorsten Lindloff

Wer das erste Mal mit einem freien Betriebssystem wie NetBSD in Berührung kommt und vorher noch nicht mit Unix oder einem Derivat zu tun hatte, dem werden wahrscheinlich mehrere Dinge auffallen. Es gibt ein Installationsprogramm, dessen Oberfläche aussieht, als sei sie aus alten DOS-Zeiten. Es ist sehr funktionell, aber hübsch kann man es nicht nennen. Es ist unter den professionellen Systemen dennoch in guter Gesellschaft, denn selbst die Installationsprogramme von teuren kommerziellen Unix-Betriebssystemen wie HP-UX sehen nicht viel bunter aus.

Diese Open-Source-Betriebssysteme sind speicherschonend. Das kennt man nicht, wenn man von Windows kommt. Für NetBSD kann man auch den alten 386-er vom Speicher holen und damit arbeiten. Auch Exoten wie eine VAX, ein Amiga oder eine alte Sparc-Station arbeiten mit NetBSD gut zusammen. Solche Rechner kann man heutzutage für ein paar Euro gebraucht erwerben und noch nutzbringend einsetzen.

Wenn Sie im Internet surfen wollen, müssen Sie die Netzwerkfunktionalitäten installieren. Unter NetBSD ist es ganz einfach, die Verbindung mit mehreren Rechnern zu teilen, also so etwas wie Routing aufzusetzen. In zwei kleinen Dateien sind mit einem Editor ein paar Modifikationen vorzunehmen und mit etwas Übung steht eine Netzwerkverbindung innerhalb von etwa zwanzig Minuten. Ohne Neustart, versteht sich.

Wenn das System installiert ist, stellt sich die Frage nach der Datensicherheit. Wenn Sie einen reellen Paketfilter aufsetzen wollen, tun Sie das unter NetBSD am besten mit *ipfilter*. Ist das System korrekt konfiguriert, arbeitet das Programm so perfekt wie so manche teuere kommerzielle Lösung.

Zu obigen Überlegungen gesellt sich – besonders für Privatleute – noch eine weitere Frage, nämlich die zu den Kosten. Wenn Sie ein System in der Leistungsklasse wie das in diesem Buch gezeigte möchten und es mit kommerziellen Programmen aufsetzen, werden Sie sehr schnell tief in die Tasche greifen müssen. Sie brauchen tatsächlich alle zwei Jahre einen neuen Computer, wenn Sie alle Updates kommerzieller Systeme mitmachen wollen. Computer sind bekanntlich nicht billig und gute kommerzielle Software ist es erst recht nicht.

Sie haben das nötige Kleingeld? Dann können Sie den alten Computer verkaufen oder wegwerfen und verlieren eine Menge Geld. Oder Sie benutzen ihn weiter. Eine Möglichkeit dafür wäre, einmal ein System wie NetBSD auszuprobieren und die Maschine nach Belieben als Router, Print- oder Fileserver oder auch für solch triviale Dinge wie die tägliche Arbeit herzunehmen.

Wenn Sie sich mit Linux auskennen, wird Ihnen beim Studium des Buchs auffallen, daß es zwischen diesem System und NetBSD etliche Parallelen gibt. Die Software ist bei beiden Systemen die gleiche. Ein Nameserver ist immer ein Nameserver, egal, was auf der CD des Distributors steht. Für Sendmail, Postfix, Apache und vieles anderes interessante, was NetBSD so zu bieten hat, gilt das analog.

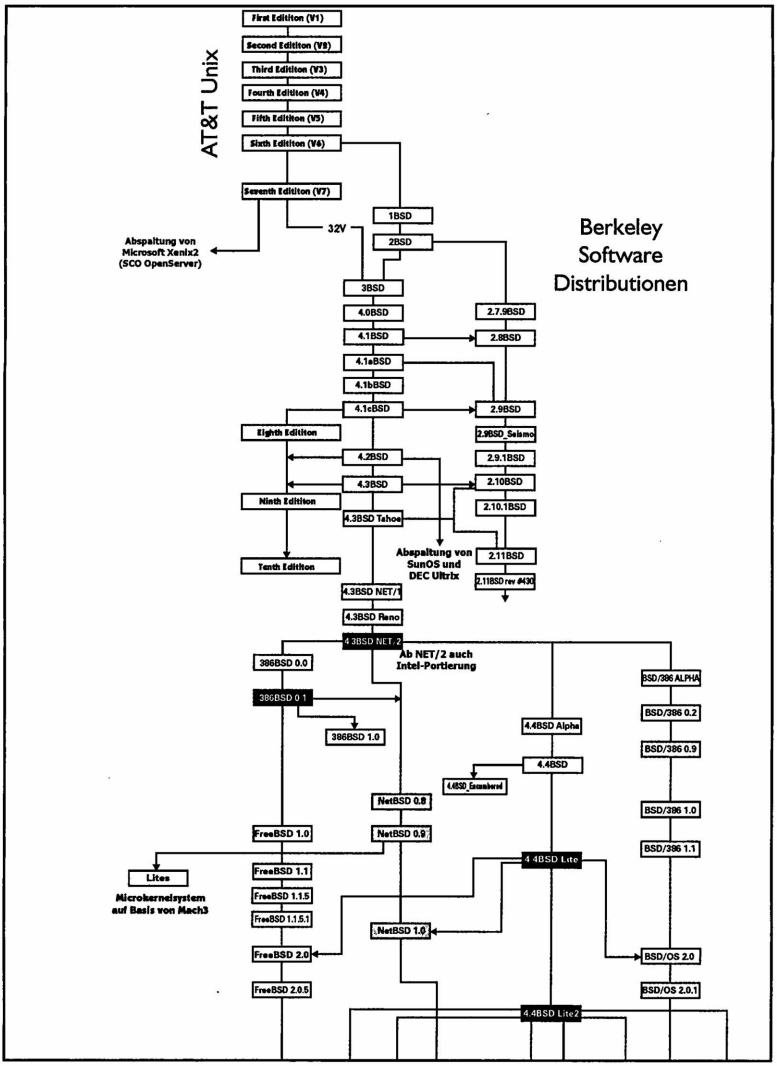


Abb. 1.1: Der Stammbaum von BSD (Teil 1). Für die Ableitung entscheidende Versionen sind schwarz, NetBSD grau hinterlegt dargestellt

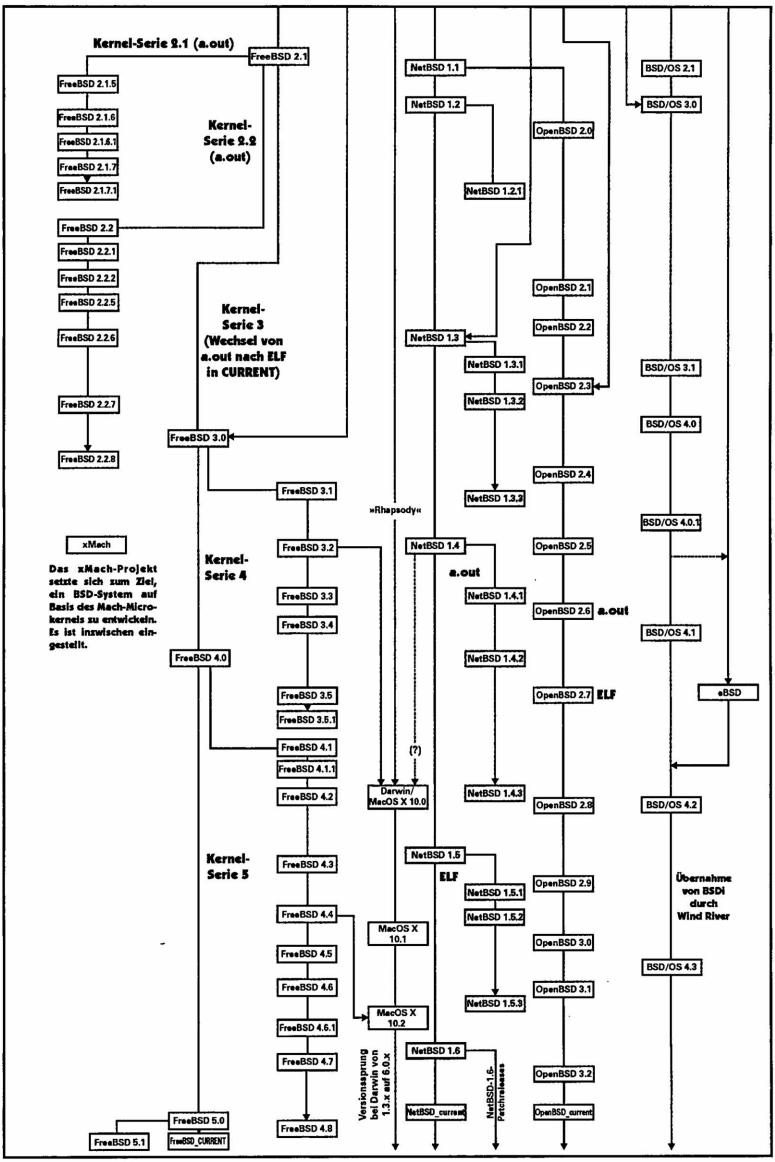


Abb. 1.1: Der Stammbaum von BSD (Fortsetzung)

Das kommt daher, daß die Wurzeln von NetBSD bis in die Anfänge von Unix zurückreichen, also bis in die Mitte der sechziger Jahre. Linux kam erst 1991 in einer ersten Version. Weil man Software dafür brauchte, wurde vieles aus dem BSD-System übernommen und nach Linux portiert. Als Resultat aus dieser langen Tradition der kontinuierlichen Weiterentwicklung bekommen Sie mit NetBSD ein wirklich ruhig und stabil laufendes System, das Sie im Prinzip einschalten und dann gar nicht mehr bemerken.

Natürlich geht es auch umgekehrt: Heute werden viele Anwendungen zuerst für Linux geschrieben und dann auf ein BSD-System portiert beziehungsweise laufen sie in einem Emulator. Zwei Beispiele dafür sind der Netscape Communicator und der Acrobat Reader von Adobe.

Mittlerweile gibt es für NetBSD auch ein komplettes Office-Paket, das ohne Linux-Emulation auskommt. Das ist OpenOffice, das aus StarOffice 5.2 hervorging und vorkompiliert auf dem FTP-Server der NetBSD-Organisation zum Download bereitsteht. Wenn Sie ein solches Päckchen selbst kompilieren wollen, müssen Sie einplanen, daß Sie dann einige Zeit einen Rechner weniger zur Verfügung haben. Die Kompilierung von OpenOffice dauert auf einem Pentium-III-Rechner mit einer Taktfrequenz von einem GHz zwanzig Stunden, wenn das System sauber konfiguriert ist und die Quelldaten einwandfrei sind. Empfehlenswert ist das Selbstkompilieren aber trotzdem.

Natürlich gibt es trotzdem große Unterschiede zwischen den Unix-Systemen. Schließlich hätte sonst dieses Buch keine Existenzberechtigung :-)

Es gibt auch nicht »das BSD-Betriebssystem«, sondern neben NetBSD noch drei weitere Varianten, nämlich FreeBSD, OpenBSD und BSDi (die kommerzielle Version).

NetBSD läuft aktuell auf rund sechzig Plattformen – vom PC bis hin zu Exoten wie dem PC532, von dem 1991 gerade einmal 197 Motherboards hergestellt wurden. Es läuft nicht auf allen Plattformen gleich komfortabel, aber auf den meisten, die auf der Website *http://www.netbsd.org/* angegeben sind, funktioniert es einwandfrei. Die wichtigsten sind:

- i386
- Apple: PowerMac und 68k
- Alpha: Die Maschinen von Compaq und DEC
- Amiga: Alles ab 68020 mit MMU, 68LC-Prozessoren laufen derzeit nicht korrekt
- Atari: TT30, Falcon und Hades
- HP300: Workstations von HP mit 68k-Prozessoren
- HPCArm: Journada und IpaQ
- HPCMips
- HPCsh
- Luna68k
- Mipsco
- Next68k
- News68k: Eine Sony-Maschine

- PC532
- Pmax: DECstations mit MIPS-Prozessoren
- VAX: Ebenfalls eine Maschine von DEC
- Sun: Sparc, Sparc64, Sun3 und Sun3

Interessant ist auch eine völlig andere Plattform: Wußten Sie, daß ein paar Leute daran arbeiten, der XBox von Microsoft zu einem NetBSD-Kernel zu verhelfen?

1.1 Wie es anfing

Die Geschichte der BSD-Systeme ist mit der von Unix eng verwoben.

Bereits 1965 arbeiteten an der ersten Entwicklung des Vorläufers von Unix namens *Multics* die Bell Labs (im Eigentum der Telefongesellschaft AT&T), das MIT und die General Electric Company. Dieses Betriebssystem sollte über Eigenschaften verfügen, die heute als selbstverständlich angesehen werden:

- Eine große Gruppe von Benutzern sollte gleichzeitig an der Maschine arbeiten können.
- Die Rechenleistung und der Speicher sollten allen Benutzern in gleichem Maße zur Verfügung stehen.
- Der einzelne Benutzer soll ohne Umschweife auf gemeinsame Daten zugreifen können.

Bis 1969 schaffte man es, eine einfache Variante dieses Betriebssystems fertigzustellen. Allerdings erfüllte es nicht die gestellten Anforderungen, weshalb die Bell Labs 1969 die Arbeit an diesem System einstellten.

Später begannen einige Mitarbeiter – darunter Ken Thompson und Dennis Ritchie – des Computer Science Research Centers der Bell Labs damit, ein Dateisystem zu entwikkeln, das eigentlich nur die Programmierumgebung verbessern sollte. Dieses Dateisystem entwickelte sich im Laufe der Zeit zur ersten Version des Unix-Dateisystems. Kernigham nannte das neue Betriebssystem *Unix* (als Verballhornung von »Multics«).

Schon die allererste Unix-Version war weiter entwickelt als es Multics jemals war. Es war zu dieser Zeit aber noch kein offizielles Projekt der Bell Labs. Trotzdem war Unix schon so weit, daß man Texte damit verarbeiten konnte und daß es in der Patentabteilung der Bell Labs bereits als Textverarbeitungssystem im produktiven Einsatz war. Es diente außerdem in Universitäten zu Forschungszwecken.

In dieser Zeit entstanden die Programmiersprachen B und als Fortentwicklung C, mit der der Unix-Kernel anschließend geschrieben wurde.

Damals erlebte der Computer auch für kommerzielle Zwecke seine erste Blütezeit. Nur AT&T hatte Pech: Wegen des damals noch gültigen Consent Decree von 1956 durfte die Firma keine Computer oder Produkte dafür vermarkten.

1976 gelangte Unix bei der Version 6 an. Diese Version wurde weit verbreitet. Mit Version 6 kamen auch neue Programme wie make, lint, awk und uucp hinzu.

Wegen seiner technischen Überlegenheit gewann Unix in der Folgezeit sehr stark an Popularität. 1977 waren etwa 500 Maschinen mit diesem Betriebssystem ausgerüstet, von denen etwa ein Viertel in Universitäten stand.

Damals entstand auch die erste kommerzielle Unix-Variante, die von der Interactive System Corporation vertrieben und um einige Office-Automationsanwendungen erweitert wurde.

Das gute Design motivierte auch damals schon viele Entwickler dazu, eigene Unix-Versionen zu entwickeln. Die ersten Portierungen auf andere Plattformen als PDP waren IBM 360 (1977) und Interdata 8/32 (1978). Es handelte sich dabei um Version 7, die 1978 auf den Markt kam. Sie bildet die Basis für nahezu alle heute verfügbaren Unix-Derivate. In dieser Version fanden Erweiterungen statt, die ein entscheidender Schritt nach vorne waren: Es gab keine Beschränkung der Benutzerzahlen mehr, außerdem waren jetzt ein kompletter C-Compiler und eine neue Benutzershell implementiert (die Bourne Shell). Das Programmierhandbuch war mittlerweile auf 400 Seiten und zwei Anhänge gewachsen, die ebenfalls jeweils an die 400 Seiten umfaßten. Der von Chuck Haley und Bill Joy in Berkeley entwickelte Editor vi floß ebenfalls in diese Version ein.

Ende der siebziger Jahre kamen auch die ersten Mikrocomputer auf den Markt. Die preiswerteren Maschinen waren für den Alltagseinsatz im Büro gedacht, die qualitativ besseren und schnelleren Maschinen, die »Workstations« genannt wurden, liefen mit Unix. Allerdings gab es von jedem Hersteller eigene Varianten, die nur begrenzt miteinander kompatibel waren. Einige dieser Betriebssysteme waren BSD-Abkömmlinge, andere basierten auf dem Quellcode von System V. Die Microsoft-Variante (Xenix) stammt von System III ab. Xenix als solches wurde von anderen Firmen wie Intel und SCO lizensiert.

Neben der Weiterentwicklung von Unix brachte das neue Release auch einen Einschnitt: Bei AT&T wurden die Quellen von Version 7 an als Geschäftsgeheimnis betrachtet. Ab diesem Zeitpunkt untersagten die Lizenzbestimmungen das freie Erläutern der Quelltexte in den Vorlesungen der Universitäten. Vorlesungen über die Betriebssystementwicklung wurden von da an auf die Theorie beschränkt.

Es gab etliche Professoren, die sich daran hielten und mindestens einen, der sich davon nicht abschrecken ließ: Andrew Tanenbaum, seines Zeichens Netzwerkspezialist an der Freien Universität in Leiden/Niederlande. Er entwickelte damals Minix.

Das BSD, das aus Version 7 portiert wurde, war 3BSD.

1979 spaltete sich Unix in drei Varianten auf: die BSD-Organisation, die kommerziellen Varianten von AT&T und das Forschungssystem.

4BSD wurde 1980 veröffentlicht, es besaß ein schnelleres Dateisystem und ein Mail-System. Diese Version war die letzte ohne Neunumerierung.

Die Software und wesentliche Teile des Kernels wurden von der BSD-Organisation entwickelt. Dennoch hatte man bis dahin noch keinen kompletten eigenen Kernel. Das System mußte zweimal lizensiert werden: es gab sowohl die freie Lizenz bei BSD und dann war noch ein Obolus an AT&T zu entrichten.

1981 wechselte Bill Joy von Berkeley zu Sun. Er nahm die Quellen von 4.1c mit und entwickelte sie dort weiter. Als 4.2BSD fertig war, übernahm Sun die Korrekturen für dieses System und entwickelte daraus ein erstes Release für das heute noch verwendete *SunOS*, das aus Marketinggründen in *Solaris* umbenannt wurde.

Aus der Version 4.1c entstanden auch einige andere Derivate wie Ultrix von DEC. Aus dem selben Grund übernahm Kirk McKusick den Job als BSD-Chefprogrammierer. Unter seinen Fittichen entstanden die Versionen 4.3 und 4.4. Er war es auch, der das Wappenwesen der BSD-Organisation, den BSD-Daemon, entwarf und bis heute die Rechte daran hält.

Ab Version 4.3 entwickelte BSD bereits ein Eigenleben mit zwei Varianten: 4.3BSD und 4.3Tahoe. 4.3BSD wurde zu 4.3BSD Tahoe weiterentwickelt, aus der 1990 Net/1 entstand.

Die achtziger Jahre waren auch die Dekade, in der als Reaktion auf die Lizenzpolitik von AT&T die FSF, Free Software Foundation, entstand. Sie wurde von Richard M. Stallman gegründet, um allen Unix-Benutzern, zumindest denen an den Universitäten, freie Software anbieten zu können.

Damals entstanden etliche Programme neu und viele dieser neuen Programme fanden nach und nach auch ihren Eingang in die BSD-Systeme. GNU-Compiler, Fortran, Pascal, ein Lisp-Interpreter, Sendmail, und ein neuer vi-Klon, nvi, sind die bekanntesten Beispiele. Nicht vergessen sollte man dabei auch den Editor Emacs, den es vorher schon für ITS gab und der mit einer Makrosprache, die Lisp ähnlich ist, programmiert werden kann. Außerdem wurde die gesamte Systemsoftware nach und nach neu entwickelt.

Gegen Ende der achtziger Jahre mauserte sich der PC zu einer preiswerten Maschine mit dem Betriebssystem MS-DOS. Die einzige grafische Oberfläche, die es für diese Rechner gab, war Windows, das in einer ersten einfachen Version erst 1983, nach dem Macintosh von Apple, erschien. Später kam OS/2 dazu, das eine Gemeinschaftsentwicklung von IBM und Microsoft war.

Damit hatte Unix zuerst einmal verloren: Das CDE-Projekt, das eine einheitliche grafische Oberfläche für Unix werden sollte, kam zu spät und war für die damaligen Rechner zu langsam und CDE war zu teuer.

Ein anderes Beispiel für ein gescheitertes Unix-Projekt sind die Next-Workstations. Next Inc. wurde von Steve Jobs gegründet, nachdem er Apple verlassen hatte. Die NextStation war damals einer der besten Computer auf dem Markt: sie besaß eine komfortable grafische Benutzeroberfläche und man konnte Grafikbeschleunigerkarten einbauen. Das Betriebssystem war eine Kombination aus dem Mach-Microkernel und dem BSD-Kernel. 1996 wurde die Firma von Apple gekauft.

Anfang der neunziger Jahre begann AT&T, seine Rechte an Unix geltend zu machen. Das war ein großes Problem für die BSD-Organisation, das gelöst wurde, indem man zwei Varianten auf den Markt brachte: 4.4BSD und 4.4BSD-Encumbered. Für letztere Version mußte eine Unix-Source-Lizenz erworben werden.

Ab 1990 begannen auch die Gerichtsprozesse um die Urheberrechte zwischen AT&T und der Berkeley Software Design Inc. (BSDi). Diese Prozesse zogen sich bis 1993 hin.

AT&T gewann und die BSD-Organisation mußte hinnehmen, daß nur noch Fragmente des BSD-Kernels in den BSDs verwendet werden konnten. 1993 verkaufte AT&T die Rechte an Unix an Novell.

1991 erschien Net/2, das gleichzeitig gleichzeitig die erste Portierung für die Intel-Plattform war.

Aus 4.4BSD wurde der lizenzpflichtige Sourcecode weitgehend entfernt und das fehlende Material durch eigenes ersetzt. 4.4BSD endete mit den Versionen 4.4Lite1 und 4.4Lite2. In 4.4Lite2 blieben noch sechs Dateien übrig, die durch eigenen Code ersetzt werden mußten.

Bill Jolitz schaffte es 1992 schließlich, BSD erstmals aus eigenem Code auf den i386-PC zu portieren. Diese Variante nannte man 386/BSD 0.1. Sie war sehr fehlerhaft, so daß es Jolitz allein nicht mehr schaffte, mit dem Entwickeln der Patches nachzukommen. Noch im selben Jahr erschien eine neue Version von 386/BSD mit der Nummer 0.5.

Aus dieser Version entstanden 1993 zwei neue Projekte: NetBSD und FreeBSD. FreeBSD konzentriert sich auf die zwei Plattformen PC und die Alpha-Prozessoren[1]. Bei ihm steht die einfache Bedienung des Systems im Vordergrund. NetBSD setzt andere Präferenzen: es soll auf möglichst vielen Plattformen laufen. Trotz der Vielseitigkeit des Systems wird auch hier großer Wert auf eine einfache Konfiguration, sorgfältige Dokumentation und die gute Bedienbarkeit des Systems gelegt. Die Manpages gelten als die besten, die es auf dem Sektor gibt.

1995 trennte sich Theo de Raadt im Streit von der NetBSD-Gruppe und startete sein eigenes BSD-Projekt: OpenBSD. Es wird genauso weiterentwickelt wie die anderen Unix-Varianten, auch die Philosophie ähnelt der von NetBSD. Der Unterschied zwischen beiden ist zum einen die weitaus stärker auf Netzwerksicherheit bedachte Entwicklung der Software und zum anderen die stärkere Orientierung auf neue Software, was sich auch in den Versionsnummern ausdrückt. OpenBSD ist bereits bei Version 3.x. Aber es ist nicht so weit portierbar wie NetBSD. Außerdem kommt das System nicht aus den USA, sondern aus Kanada, um die damals recht restriktiven Exportbestimmungen der Vereinigten Staaten für Verschlüsselungssoftware zu umgehen.

Das NetBSD-Projekt ist seit seiner Entstehung 1993 ein internationaler Zusammenschluß etlicher Programmierer, die ein frei verfügbares und verteilbares Betriebssystem schaffen wollten. Weitere Ziele sind die Kompatibilität zu möglichst vielen Programmierschnittstellen, Binärkompatibilität zu anderen Unix-Systemen und die Unterstützung möglichst vieler Netzwerkprotokolle.

Die NetBSD-Gruppe nimmt eine sehr konservative Haltung hinsichtlich neuer Software oder irgendwelcher Features ein. Denn wenn man etwas solides hat, baut man nur sehr vorsichtig aus, wenn man nichts dabei verlieren will. Das drückt sich bei NetBSD auch in den Versionsnummern aus. Erst 1995 konnte man sich zu einer 1.0-Version durchringen. Mittlerweile ist man bei Version 1.6 angekommen.

---

[1] Die neue Version 5.0 von FreeBSD gibt es außerdem auch für ia64 und Sparc64

2 NetBSD installieren

NetBSD ist auf zahlreiche Hardware-Plattformen und verschiedene Prozessoren portiert. Die Installation auf über vierzig verschiedenen Plattformen zu beschreiben, unter denen sich auch echte Exoten-Hardware befindet, von der in manchen Fällen nur sehr wenige Exemplare produziert wurden, ist unmöglich. Außerdem ist im Installationsverzeichnis jeder Plattform eine knappe (englische) Installationsanweisung verfügbar. Diese Beschreibung ist im Postscript, HTML- und Textformat abgelegt und deckt die wichtigsten Installationsschritte ab. Allerdings wird in diesen Beschreibungen auf Fallstricke oft nicht ausreichend hingewiesen

Um auch NetBSD-unerfahrenen Anwendern den Einstieg in das Betriebssystem zu erleichtern, sind in diesem Kapitel beispielhaft die Installationen auf Intel, Macintosh (68000-Familie, kurz »mac68k«), dem Amiga und der DECstation sowie auf den 32- und 64-bittigen Sparc-Maschinen von Sun beschrieben. Es werden auch alternative Installationswege gezeigt und Hürden und ihre Beseitigung diskutiert.

2.1 Die Installation auf dem Intel-PC

von Thorsten Lindloff

Auch wenn ein solcher Abschnitt manchem vielleicht überflüssig erscheinen mag, ist es wichtig, die Beschaffungswege von NetBSD vor der Installation zu klären. Gerade bei einem Betriebssystem, dessen Entwicklung und Verbreitung primär über das Internet stattfindet, sind die Beschaffungswege oft identisch mit den Installationswegen. Es gibt einige Methoden, an das notwendige Material zu kommen:

1. Die Verwendung der CD-ROM beispielsweise von einer Ausgabe der freeX. NetBSD 1.6 für die Intel-Plattform befand sich auf der CD-ROM der Ausgabe 6/2002.
2. Der Erwerb eines kompletten CD-Sets. Unter *http://www.netbsd.org* lassen sich die Adressen von Distributoren der jeweils aktuellen Version erfahren.
3. Download über das Internet via ftp.
4. Eine Kombination der angesprochenen Beschaffungswege.

2.1.1 Die Installation von CD-ROM

Wenn Ihnen eine CD-ROM vorliegt, ist der Start der Installation kein Problem. Sie müssen nur die CD-ROM ins Laufwerk legen, eventuell die Bootreihenfolge ändern und den Installationsanweisungen folgen.

Sollte das Booten direkt von CD-ROM nicht funktionieren, was bei älteren Maschinen oft der Fall ist, müssen Sie sich ein paar Bootdisketten bauen.

Beim Download via ftp stehen Ihnen zwei Möglichkeiten offen. Der Hauptserver *ftp.netbsd.org* stellt im Verzeichnis »/pub/NetBSD/iso« Boot-Images für den Download zur Verfügung. Solche Images gibt es für nahezu jede Plattform. Es muß also nur das

entsprechende Image heruntergeladen und mit einem beliebigen Brennprogramm auf eine CD-R übertragen werden. Diese Variante ist sehr einfach zu handhaben, sie besitzt aber den Nachteil, daß in jedem Fall das komplette Installationsset heruntergeladen werden muß.

Der Aufbau einer CD-ROM für die Installation von NetBSD/i386

Das Schema des Aufbaus ist immer gleich. Die Dateien werden immer in diesen Verzeichnissen erwartet. Diese Pfade werden auch als Voreinstellung angeboten. Stimmen sie, läuft die Installation am problemlosesten ab:

/i386/binary/kernel	Hier sollten die diversen Kerneldateien als Archive abgelegt sein.
/i386/binary/sets	Hier **müssen** die Sets für die Installation sein.
/i386/binary/installation/floppy	Hierhin gehören die Installations-Floppy-Images.
/i386/binary/utilities	Das DOS-Boot-Utility.
/i386/utilites/	Hilfsprogramme (*rawrite.exe* und *pfdisk.exe*).
/source	Der passende Platz für die System- und X-Quellen.
/packages	Der Ort, an den Binär-Packages der Fremdsoftware gehören.

Wenn Sie eine CD-ROM mit diesen Daten brennen, müssen Sie *mkisofs* den Pfad zum El-Torito-Bootimage angeben. Das ist die Datei »i386/binary/installation/floppy/boot-big.fs«. Ein Aufruf von *mkisofs* sollte also wie folgt aussehen:

```
mkisofs -A "NetBSD 1.6" -b "i386/installation/floppy/boot-big.fs" \
        -c "i386/installation/floppy/boot.catalog" -d -D -L \
         -J -P "Ihr Name und E-Mail" -p "Ihr Name und E-Mail" -r -R \
         -V "NetBSD-Install" -o /pfad_zum_iso_image/netbsd.iso \
          /Quellpfad
```

Für erfahrene Anwender sind auf dem Server auch die Pakete einzeln herunterladbar. Das macht vor allem dann Sinn, wenn die Installation beispielsweise über ein lokales Netz vorgenommen werden soll oder nur ein Teil der Grundpakete benötigt wird. Warum soll man auch auf einem Server die ganzen Pakete für die grafische Installation oder die BSD-Spiele installieren? Mit diesen Paketen ist auch die Installation direkt aus dem Internet möglich, es wird nur eine Bootdiskette benötigt. Diese Art der Installation setzt aber auf jeden Fall eine schnelle Internetanbindung voraus.

Die günstigste Lösung für den Normalanwender stellt aber in jedem Fall der Erwerb eines vorkonfektionierten CD-Sets dar. Auf der NetBSD-Website gibt es eine Anbieterliste. Sie können zum einen die Originale von verschiedenen Distributoren erwerben und zum anderen auch eigene Versionen verschiedener Anbieter. Die gibt es mittlerweile auch in Deutschland.

Gehen wir davon aus, daß Sie die einfachste Art der Installation wählen und der Rechner ein CD-ROM-Laufwerk eingebaut hat. Das ist zwar keine Grundbedingung, die Installation ist auch über das lokale Netz entweder per ftp oder nfs möglich, eine solche Installation wird aber kaum die Regel sein.

Die Installations-CDs von NetBSD für die Intel-Plattform sind bootfähig nach dem El-Torito-Standard. Das heißt, auf die CD-ROM ist ein Diskettenimage gebrannt, das beim Systemstart genau wie ein Diskettenlaufwerk bootet. Dieses Image hat eine Größe von 2,88 Mbyte, was auch auf normalen Systemen, die per Hardware so ein Laufwerk nicht besitzen, keine Hürde darstellt. Damit das System von diesem Image bootet, muß bei IDE-Systemen im BIOS die Bootreihenfolge, die normalerweise *Festplatte C->Diskette* oder *Festplatte1->Festplatte2* ist, umgestellt werden. Primäres Boot-Device muß das CD-ROM-Laufwerk sein. Im BIOS ist dafür in den erweiterten Option die Reihenfolge auf *CD-ROM->C* umzustellen.

Bei SCSI-Systemen sieht es etwas anders aus. Das Booten von CD ist generell auf SCSI-Systemen nur möglich, wenn der SCSI-Controller ein eigenes BIOS besitzt. Billig-Controller oder auch sehr alte Controller, die gern in 486er-Maschinen eingesetzt wurden, haben in der Regel keines. Auch bei SCSI-Geräten muß im Rechner-BIOS die Bootreihenfolge geändert werden, damit von der CD gestartet werden kann, allerdings so, daß von Diskette gestartet wird. Das heißt: bei SCSI-Geräten wird nicht nominell von CD gebootet, sondern von einer Diskette, die dem System vorgegaukelt wird. Anschließend muß im Controller-BIOS die Unterstützung für bootbare CDs eingeschaltet werden. Der Umweg ist deshalb nötig, weil das SCSI-BIOS dem Rechner-BIOS vorgeschaltet ist und dort keine Eingriffe vornimmt. Das SCSI-BIOS gaukelt dem normalen BIOS eine echte Bootdiskette im ersten Diskettenlaufwerk vor und das Rechner-BIOS bootet dann davon. Das Original-Diskettenlaufwerk wird bei dieser Aktion übrigens umgemappt und steht als zweites Diskettenlaufwerk immer noch zur Verfügung.

Leider bootet aber nicht jeder PC von einer CD-ROM. Ältere Maschinen können es generell nicht, da zur Herstellungszeit des BIOS der El-Torito-Standard noch nicht definiert war, neuere Maschinen mit fehlerbehaftetem BIOS oder älteren CD-Laufwerken machen oft Probleme, insbesondere beim Booten von CD-Rs. Auch sind Rechner bekannt, die große Schwierigkeiten beim Booten von Images haben, die größer als 1,44 Mbyte sind. In diesem Fall muß für die Installation ein Diskettenset angefertigt werden.

Nehmen wir beispielsweise einen 486DX2-66, auf dem das Booten von CD nicht möglich ist. Das System muß von Disketten gebootet werden. Zuerst muß natürlich bekannt sein, auf welchem Betriebssystem die Disketten geschrieben werden sollen. Das klingt wie eine Binsenweisheit, ist aber für die Vorgehensweise wichtig. Fangen wir mit dem prominentesten Fall an. Auf der Maschine ist ein DOS-basiertes Windows installiert, dabei spielt es keine Rolle, ob Windows 3.x auf DOS 6.22 oder Windows 95.

El-Torito

Beim El-Torito-Standard für bootbare CDs befindet sich auf dem Datenträger das Eins-zu-Eins-Image einer Diskette. Dieses Image muß entweder 1,2, 1,44 oder 2,88 Mbyte groß sein. Andere Diskettengrößen werden nicht unterstützt. Wenn im BIOS die entsprechende Unterstützung eingeschaltet ist, versucht das System, dieses Image zu starten. Außer dem Image selbst wird zwingend auch ein Boot-Katalog benötigt. Er wird von Mastering-Programmen wie *mkisofs* und *mkhybrid* angelegt. Der El-Torito-Standard definiert nicht, wie das System die CD-ROM bootet, sondern nur den Aufbau der bootfähigen CD-ROM. *mkisofs* unterstützt auch mehrere (bis zu 16 Bootimages) auf einer CD-ROM. Das zu bootende Image wird dann über ein Startmenü ausgewählt. Die CD-ROM NetBSD Live! zu diesem Buch enthält zwei solcher Bootimages.

Um die beiden Startdisketten zu erzeugen, wird die CD ins Laufwerk gelegt. Nun öffnen Sie ein DOS-Fenster und legen die erste formatierte und unbedingt fehlerfreie Diskette ein. Auf der CD müssen die folgenden Verzeichnisse und Dateien vorhanden sein:

```
d:\i386\installation\floppy:
boot1.fs
boot2.fs

d:\i386\installation\misc:
rawrite.exe
```

Nehmen Sie ein *cd* in das Verzeichnis »i386\installation\misc« vor und rufen Sie das Programm *rawrite.exe* auf. Es reicht, wenn Sie an dieser Stelle nur den Programmnamen angeben. Sie werden anschließend nach dem Pfad und Namen des Diskettenimages und nach dem Ziellaufwerk gefragt. Sie müssen hier den Pfad mit Slashes angeben. *rawrite* verwendet die Unix-Slashes und nicht den Backslash von DOS:

```
/i386/installation/floppy/boot1.fs
```

Jetzt wird diese Datei auf die Diskette geschrieben. Vorher werden alle Daten auf der Diskette wegen einer Neuformatierung gelöscht. Sollten Sie gebrauchte Disketten verwenden, stellen Sie bitte unbedingt sicher, daß keine wichtigen Daten darauf abgelegt sind. Beschriften Sie diese Diskette mit »NetBSD Boot 1« oder ähnlich, damit Sie wissen, was sich darauf befindet. Diesen Vorgang wiederholen Sie mit der Datei »boot2.fs«.

Das war bereits der erste Schritt für DOS-/Windows-Benutzer. Unter Windows NT, Windows 2000 und Windows XP funktioniert das Programm *rawrite.exe* nicht. Für das Schreiben der Diskettenimages muß dort das Programm *rawrite32.exe* verwendet werden. Es handelt sich dabei um eine Windows-GUI-Applikation (Abb. 2.1).

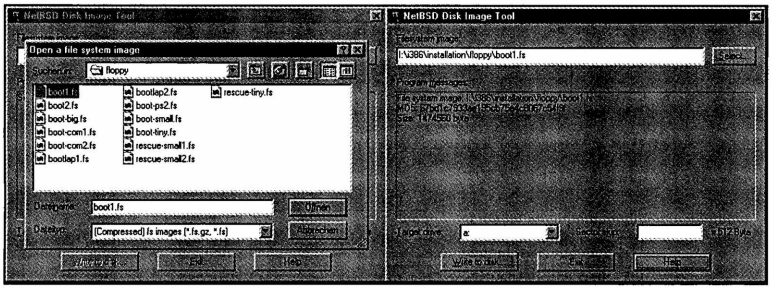


Abb. 2.1: Das GUI-Programm »rawrite32.exe« für Windows zum Erzeugen der Diskettenimages. Links die Auswahl der Boot-Images, rechts die Angaben zur ausgewählten Datei

Will man die Disketten unter einem Unix-Derivat anlegen, geht man anders vor. Zuerst wird die CD-ROM gemountet, hier beispielsweise unter NetBSD selbst nach »/mnt«:

```
#mount_cd9660 /dev/cd0d /mnt
```

Alternativ geht anstelle von *mount_cd9660* auch *mount -t cd9660*. Hier ist die Prüfung mit *ls /mnt/i386/installation/floppy*, ob es sich um die richtige CD-ROM handelt, genauso obligatorisch. Dann wird das Image auf die Diskette geschrieben:

```
#dd if=/mnt/i386/installation/floppy/boot1.fs of=/dev/fd0a
```

Die Datei »boot1.fs« wird jetzt auf das Device »/dev/fd0a«, das die direkte Geräteangabe für das ersten Diskettenlaufwerk im System ist, kopiert. Genauer gesagt: es wird ein komplettes Dateisystem mit einem Kernel auf die Diskette geschrieben. Der gleiche Vorgang wird anschließend mit der Datei »boot2.fs« wiederholt.

Unter Linux sind die Befehle zum Beschreiben der Diskette identisch, die Gerätenamen lauten aber anders.

Zuerst wird die CD-ROM eingebunden:

```
#mount -t iso9660 /dev/cdrom /[CD-ROM-Verzeichnis]
```

Nach dem Einlegen der leeren Diskette ist ebenfalls *dd* aufzurufen:

```
#dd if=/mnt/i386/installation/floppy/boot1.fs of=/dev/fd0
```

Auch hier müssen natürlich beide Diskettenimages erzeugt werden.

Damit kann jetzt die Installation begonnen werden.

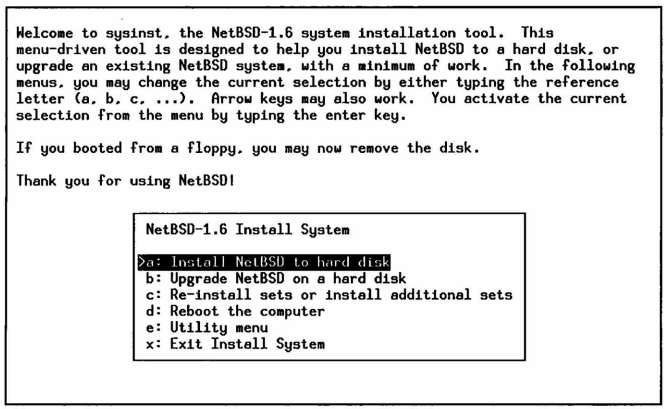


Abb. 2.2: Das Hauptmenü von sysinst

Starten Sie das System, entweder von der CD-ROM oder von der soeben erzeugten ersten Diskette. Nach einiger Zeit sehen Sie beim Booten von Diskette die Meldung, daß nun die zweite Bootdiskette eingelegt werden soll. Wechseln Sie dann die Diskette und drücken Sie die [Enter]-Taste. Nach einigen Kernelmeldungen sollte das Installationsprogramm *sysinst* gestartet werden.

Ist das nicht der Fall, ist der Rechner grundsätzlich nicht kompatibel zu dem vorbereiteten NetBSD-Kernel. Der Rechner hängt dann und beim Booten ist kein Fortschritt mehr zu sehen. Dies sollte auf Rechnern mit Standard-Hardware nicht vorkommen, speziell für Laptops[1] sind aber auf den normalen CDs alternative Installationsdisketten-Images vorbereitet. Sie müssen jetzt diese Images ausprobieren. Sie befinden sich im selben Verzeichnis wie die normalen Bootdisketten-Images und besitzen die Namen »bootlap1.fs« und »bootlap2.fs«. Diese Images sollten eigentlich nur auf speziellen Laptops benötigt werden. Die Erfahrung hat aber gezeigt, daß durchaus auch Desktop-Geräte, die Schwierigkeiten mit Treibern im normalen Kernel haben, sich damit erfolgreich starten lassen.

Im Normalfall treten aber solche Probleme nicht auf und auf einem AMD K6-2/500 darf man sich nicht wundern, wenn das Betriebssystem nach etwa dreißig Minuten soweit betriebsbereit ist, daß die Tastaturbelegung korrekt ist, das Netzwerk läuft und mindestens rudimentäres Arbeiten möglich ist. Aber selbst wenn Sie eine halbe Nacht dazu brauchen: es lohnt sich. Und es geht eben auch mit alten Rechnern, die man ansonsten nicht mehr einschalten würde.

Bei dieser Beispielinstallation wird NetBSD neu installiert. Wählen Sie Punkt *a* im Menü (Abb. 2.2). Dann werden Sie gefragt, ob es weitergehen soll. Wählen Sie hier die Option *b*, also *yes* (Abb. 2.3).

[1] Insbesondere IBM Thinkpads benötigen einen Spezialkernel

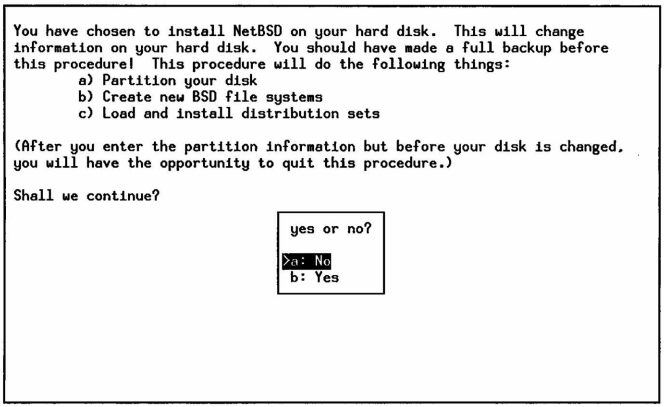


Abb. 2.3: Die Warnmeldung vor dem Beginn der Installation

Es erscheint das nächste Menü. Dort wir Ihnen mitgeteilt, wie viele Festplatten *sysinst* in Ihrem System gefunden hat. Die Meldung am oberen Rand des Bildschirms liest sich, wenn nur eine Festplatte gefunden wurde, so: »I found only one disk, wd0. Therefore I assume that you want to install NetBSD on it«. Hier ist die Auswahl eher bescheiden: *Ok*. Sie brauchen nur die Eingabetaste zu drücken.

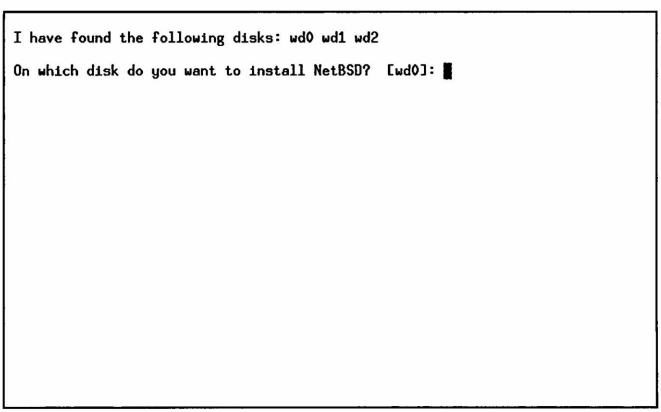


Abb. 2.4: Die Auswahl der Installationsfestplatte, wenn mehrere Platten gefunden wurden

Befinden sich mehrere Festplatten im System, muß die Auswahl für die Installationsfestplatte getroffen werden (Abb. 2.4). Dann erscheint die nächste Meldung, in der die BIOS-Werte der Platte angezeigt werden. Diese müssen nicht korrekt sein. Außerdem erscheinen wieder zwei Auswahlpunkte (Abb. 2.5).

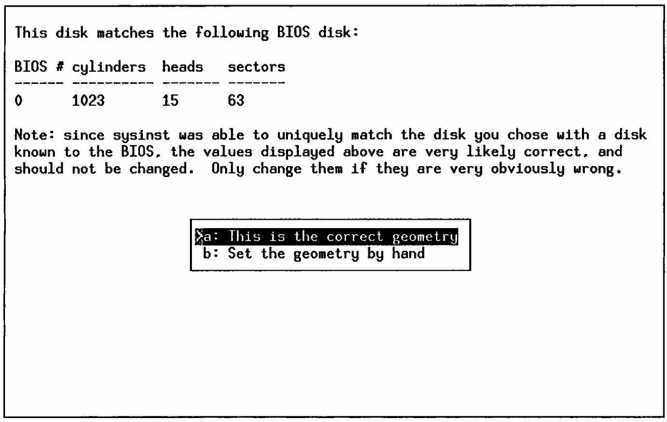


Abb. 2.5: Die Auswahl der Festplattengeometrie

Punkt *a* besagt nichts anderes, als daß die gefundene Geometrie korrekt ist. Punkt *b* fragt, ob Sie die Geometrie der Platte manuell setzen wollen. Der erste Versuch soll schnell gehen. Also nehmen Sie Option *a*.

Beachten Sie, daß gerade bei großen Festplatten, die auch noch andere Betriebssysteme enthalten sollen, hier äußerste Vorsicht geboten ist. Es kann durchaus passieren, daß durch das Verstellen der Geometrie die anderen Betriebssysteme nicht mehr starten. Wenn Sie NetBSD allein auf einer Festplatte installieren, treten solche Probleme nicht auf. Sie sollten hier aber keineswegs herumexperimentieren. Abb. 2.6 zeigt was passiert, wenn Werte eingetragen werden, die NetBSD nicht mag.

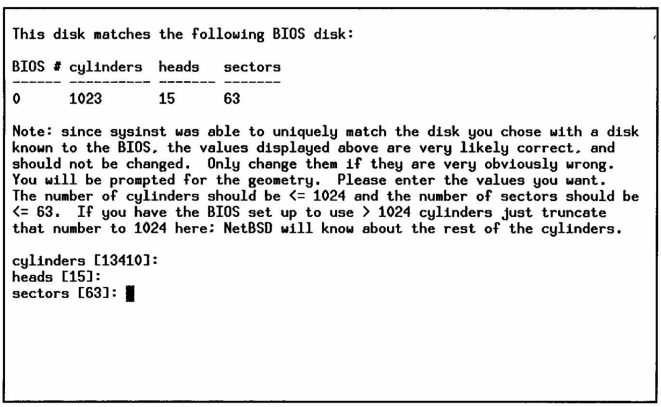


Abb. 2.6: Die Anzeige von 1023 Zylindern in Abb. 2.5 hatte durchaus ihren Sinn

Wenn Sie solche fehlerhaften Angaben machen, sollten Sie *sysinst* einfach abbrechen und von vorn beginnen. Drücken Sie dazu [Ctrl]-[C]. Das Programm wird dann beendet. Sie starten es durch die Eingabe des Programmnamens auf dem anschließend angezeigten Kommandozeilenprompt neu.

An dieser Stelle wird Ihnen auch auffallen, daß Sie »szsinst« eintippen müssen. Während der gesamten Installationsprozedur und auch nach dem ersten Start ist eine US-amerikanische Tastaturbelegung aktiv.

Im nächsten Dialog werden Sie gefragt, ob Sie die komplette Festplatte oder nur einen Teil für NetBSD nutzen wollen.

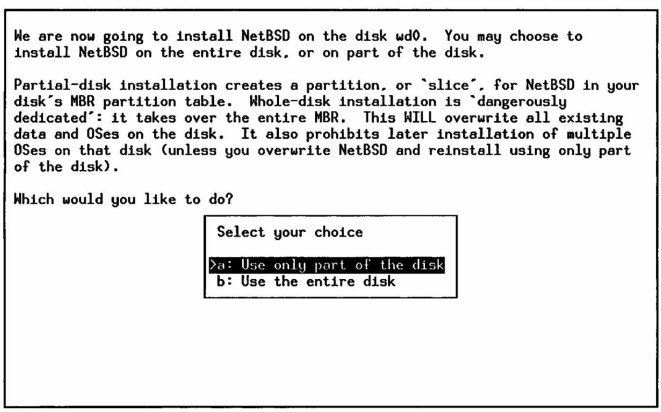


Abb. 2.7: Die Installationsoptionen

In den meisten Fällen werden Sie hier die Voreinstellung *a: Use only part of the disk* nehmen. Wenn Sie außer NetBSD keine anderen Betriebssysteme installieren wollen, sollten Sie Punkt *b* wählen. Beachten Sie in diesem Fall aber, daß kein regulärer Master Boot Record angelegt wird. Sie können dann andere Betriebssysteme nur noch installieren, wenn Sie NetBSD komplett entfernen oder komplett neu installieren und dann hier den Menüpunkt *a* wählen. Wenn Sie hier bestätigen, gelangen Sie zu einem Hinweisbildschirm, auf dem Sie aussuchen können, ob die Größenangaben bei der Partitionierung bequem in Mbyte oder in den direkten Größenangaben erfolgen soll (Abb. 2.8). Auf modernen Maschinen sollte man die Arbeit hier nicht unnötig erschweren und den voreingestellten Menüpunkt *a* wählen.

Die Auswahl *a: Megabytes* ist die einfachste, aber nicht immer die präziseste Lösung. Die Größen werden so angepaßt, daß es mit den Zylindern und Sektoren der Platte paßt. *b: Cylinders* ist weitaus präziser, aber rechenaufwendiger als die Variante mit den Mbytes. *c: Sectors* ist das genaueste Verfahren. Ein Sektor ist in NetBSD standardmäßig immer 512 Byte groß. Das heißt, Sie brauchen pro 1024 Kbyte (=1 Mbyte) genau 2048 Sektoren, um an eine passende Partitionsgröße zu kommen. Und vielleicht einen Taschenrechner, um sich die Arbeit einfacher zu machen.

Nehmen wir an, Sie haben eine Platte, die 2048 Mbyte groß ist, also zwei Gbyte. Sie brauchen bei 64 Mbyte Arbeitsspeicher einen Auslagerungsbereich, der mindestens doppelt so groß ist wie der Arbeitsspeicher, 128 Mbyte. Das wird so berechnet:

```
2048 (Sektoren/Mbyte)*128 = 262144 Sektoren
```

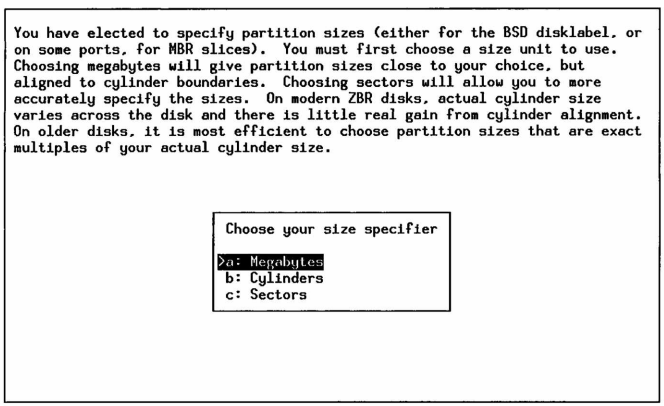


Abb. 2.8: Die Auswahl der Größenangaben

Wenn Sie die Angabe »Megabyte« wählen, rechnet das Installationsprogramm die ungefähren Werte um. Bei der ungenauen Rechnung können Bereiche auftreten, die nicht nutzbar sind. Wenn Sie mit Sektoren und Zylindern arbeiten, nimmt Ihnen *sysinst* Umrechnungen nicht ab. Partitionen müssen aber immer an Zylindergrenzen enden und beginnen, damit sie überhaupt angesprochen werden können. Wenn Sie mit Sektoren und Zylindern arbeiten, müssen Sie dieses Problem berücksichtigen. Sie müssen mit den exakten Daten der Platte rechnen, hin- und hermultiplizieren und erhöhen das Risiko einer nicht funktionsfähigen Installation nur dadurch, daß Sie es besonders gut meinten.

Zuerst einmal ein paar Erläuterungen zur Plattengeometrie:

Es gibt zwei Arten von Festplattengeometrien, die in einem Verhältnis zueinander stehen: die reale und die BIOS-Geometrie. Die reale Geometrie ist das, was auf der Platte an physikalischem Platz vorhanden ist, errechnet aus der Anzahl der Köpfe, der Anzahl der Zylinder und der Anzahl der Blöcke pro Zylinder.

Ein Zylinder wird so definiert: Eine Festplatte war in früheren Jahren eigentlich eine Sammlung mehrerer Magnetplatten, die übereinander auf eine rotierende Spindel montiert wurden. Jede dieser Magnetplatten besitzt einen Schreib-/Lesekopf, der von einem Motor über die Platte geführt wird. Daraus ergibt sich, daß alle Köpfe sich immer auf der selben Position und damit in der selben Spur auf der jeweiligen Platte befinden. Ein Zylinder ist die Summe aus der Anzahl dieser Spuren. Das bedeutet: Wenn eine Festplatte pro Spur 1200 Sektoren besitzt, enthält ein Laufwerk, das aus acht einzelnen Platten besteht, die beidseitig beschichtet sind, 19200 Blöcke pro Zylinder.

Das ist bei den heutigen Platten nicht mehr ganz so. Eine modernes Festplattenlaufwerk besteht in der Regel nur noch aus ein bis zwei Platten mit entsprechend relativ vielen Blöcken pro Zylinder und weitaus mehr Zylindern. Damit diese Zylinder von den Betriebssystemen erkannt werden konnten, ersann man irgendwann das Mapping der Zylinder und Blöcke in der Hinsicht, daß es für das Betriebssystem so aussieht, als ob nie mehr als 1024 Zylinder, aber entsprechend mehr Köpfe vorhanden sind. Genannt wird das *LBA* (Large Block Addressing).

sysinst gibt am Bildschirm immer die reale Geometrie aus, wenn die Platte weniger als acht Gbyte groß ist. Wenn eine Platte eingebaut ist, die größer ist, läuft aber trotzdem alles korrekt.

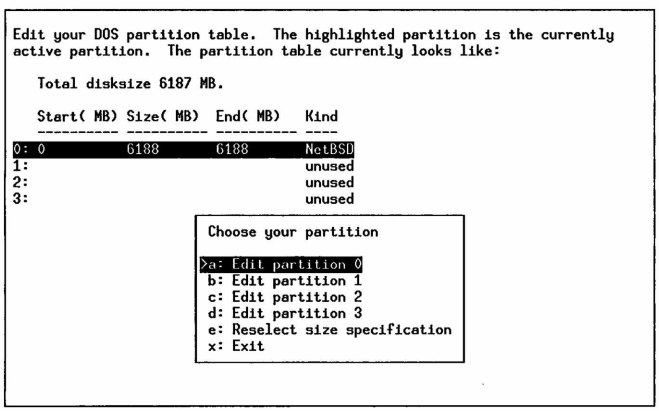


Abb. 2.9: Das fdisk-Menü von sysinst nach dem Aufruf

Es ist sehr wahrscheinlich, daß sich auf der Festplatte noch andere Partitionen befinden. *sysinst* fragt Sie, ob es weitermachen und diese Partitionen überschreiben soll oder ob Sie es bevorzugen, wenn das Programm das sein läßt. In diesem Fall kann das Betriebssystem natürlich nicht installiert werden und *sysinst* bricht ab.

Wenn Sie sich sicher sind, daß NetBSD installiert werden soll – entweder in einer neuen Partition oder auch durch Überschreiben einer vorhandenen – bestätigen Sie hier. Sie landen dann automatisch im *fdisk*-Menü (Abb. 2.9 und Abb. 2.10).

Das *fdisk*-Menü erlaubt das Löschen, Anlegen und Ändern von Festplattenpartitionen. Abb. 2.9 zeigt das Menü direkt nach dem Aufruf. Hier ist eine Partition bereits vorher angelegt und es ist passend eine alte NetBSD-Partition, die sich über die komplette Festplatte ersteckt.

Dies ist normalerweise natürlich nicht der Fall. Sie müssen sich also eine Partition aussuchen, im Untermünü *Choose your Partition* (Abb. 2.9) diese Partition mit den Cursortasten anwählen und dann [Enter] drücken. Im nächsten Bild erscheinen dann die Optionen zu dieser Partition. Die Abbildungen 2.10 und 2.11 zeigen, daß hier im Regelfall einfach der Punkt *a: NetBSD* ausgewählt werden muß. Sobald Sie hier angewählt und

bestätigt haben, wird die neue Kennung im Hauptfenster angezeigt. Sie können sie jetzt noch aktiv setzen, damit das Betriebsystem beim nächten Systemstart gebootet wird (Punkt c) und bestätigen zum Schluß mit [d] [Enter]. Sie können im gleichen Menü, falls Sie noch freien Platz haben, die Partition auch neu definieren, indem Sie mit dem Punkt b den Startpunkt (falls vorher noch Luft ist) und die Größe (falls hinter der Partition noch Platz wäre) neu bestimmen.

Sind alle Angaben getätigt und ist das Untermenü verlassen worden, wird im Hauptmenü mit [x] fdisk beendet.

Der Bootsektor von PCs erlaubt generell nur vier Partitionseinträge. Im Bootsektor sind diese mit der Größe, der Startposition und einer ID vermerkt. NetBSD-Partitionen besitzen generell die ID A9h (was dem Dezimalwert 196 entspricht). Die ID ist ein Bytewert und kann einen Wert zwischen 0 und FFh (255 dezimal) annehmen. Mit einigen Überschneidungen verwendet jedes Betriebssystem eigene IDs. Microsoft hat sich allerdings für DOS eine ganze Reihe von Kennungen erdacht. Im Master Boot Record (MBR) eingetragene Partitionen werden als »Primäre Partitionen« bezeichnet. Man stellte mit dem Größerwerden der Festplatten fest, daß vier solche primäre Partitionen oft nicht ausreichen. Dann wurde mit MS-DOS 4 das »Laufwerk in der erweiterten Partition« erfunden. Solche Laufwerke sind in einer Kette zusammengefügt. Die erweiterte Partition selbst ist aber ebenfalls im Master Boot Record eingetragen und damit auch eine primäre Partition.

Kennung hexadez./dezimal		Dateisystem(e) beziehungsweise Betriebssystem(e)
00h	0	Leere Partition
01h	1	DOS FAT 12
02h	2	Xenix, /root-Dateisystem
03h	3	Xenix, /usr-Dateisystem
04h	4	DOS FAT 16
05h	5	Erweiterte DOS-Partition
06h	6	BigDos FAT (ab DOS 4)
07h	7	HPFS (OS/2), NTFS (Windows NT/2000), QNX, Advanced UNIX
08h	8	IBM AIX /root-Dateisystem
09h	9	IBM AIX Boot-Partition
0Ah	10	OS/2 Boot-Manager
0Bh	11	Windows 95/98/2000 VFAT32 bis 2 Gbyte

Tab. 2.1 (Fortsetzung auf nächster Seite)

Kennung hexadez./dezimal		Dateisystem(e) beziehungsweise Betriebssystem(e)
0Ch	12	Windows 95/98/2000 VFAT32 LBA größer 2 Gbyte
0Eh	14	Windows 95 FAT12
0Fh	15	Windows 95 Extended
10h	16	Opus
11h	17	Versteckte FAT12 (Original-ID + 10h)
14h	20	Versteckte FAT16 (Original-ID + 10h)
16h	22	Versteckte FAT16 (Original-ID + 10h)
17h	23	Versteckte NTFS/HPFS (Original-ID + 10h)
1Bh	11	Versteckte VFAT32 (Original-ID + 10h)
1Ch	12	Versteckte VFAT32 LBA (Original-ID + 10h)
2Ah	42	AtheOS
40h	64	Venix 80286
4Dh	77	QNX Realtime Platform (erste QNX-Partition)
4Eh	78	QNX Realtime Platform (zweite QNX-Partition)
4Fh	79	QNX Realtime Platform (dritte QNX-Partition)
51h	81	Novell (?)
52h	82	Digital Research CP/M 86 und Microport System V/AT
63h	99	Unix System V, UnixWare, SCO OpenServer, GNU Hurd, Mach
64h	100	Novell Netware 2
65h	101	Novell Netware 3
75h	117	PC/IX
80h	128	Minix bis Version 1.4a
81h	129	Minix ab Version 1.4b
82h	130	Linux Auslagerungspartition
83h	131	Linux Ext2-Dateisystem, Debian GNU/Hurd
85h	133	Linux Extended

Tab. 2.1 (Fortsetzung auf nächster Seite)

Kennung hexadez./dezimal		Dateisystem(e) beziehungsweise Betriebssystem(e)
93h	147	Amoeba Dateisystem
94h	148	Amoeba Bad-Block-Table
A5h	165	BSD/386, FreeBSD, *NetBSD bis Version 1.3*
A6h	166	OpenBSD
A7h	167	NeXTStep
A9h	*169*	*NetBSD ab Version 1.4*
B7h	183	BSDi-Dateisystem
B8h	184	BSDi Swap-Partition
DBh	219	Digital Research Concurrent DOS und Concurrent CP/M
E1h	225	DOS Access
E3h	227	DOS Readonly-Partition
EBh	235	BeOS
F2h	242	Compaq DOS 3.3+ zweite Partition
FFh	255	Bad Track Table

Tabelle 2.1: Partitions-Kennungen verschiedener Betriebssysteme

Alle BSD-Systeme müssen im Gegensatz zu Linux immer in einer primären Partition installiert werden. Im Gegensatz zu Linux gehen sie aber nicht so verschwenderisch mit den Partitionen um. BSD benötigt immer nur eine Partition, die »Slice« genannt wird. Um diesen Slice zu strukturieren, werden dann die BSD-Partitionen angelegt, die in das Disklabel eingetragen werden. Bis NetBSD 1.5.3 erlaubte das Disklabel von NetBSD maximal acht Einträge im Disklabel, ab Version 1.6 sind wie bei OpenBSD sechzehn Einträge erlaubt. Erschwerend kommt hinzu, daß bei NetBSD und OpenBSD im Gegensatz zu FreeBSD, in dem Fremdpartitionen über die Slice-ID gemountet werden, auch Fremdpartitionen ins Disklabel eingetragen werden müssen. Die maximale Zahl der Einträge kann deshalb nicht groß genug sein. Die Einschränkung bis NetBSD 1.5 mit acht Einträgen bereitete große Schwierigkeiten.

Sie brauchen sich bei der Einrichtung des BSD-Slice noch keine Gedanken machen, wie groß die einzelnen Bereiche im Dateisystem sein sollten. Da im Gegensatz zum Beispiel zu Linux alles in eine Gesamtpartition kommt, gilt hier nur das altbewährte Motto »Viel hilft viel«. Die Zuordnung erfolgt erst anschließend im Disklabel-Editor.

Partition	NetBSD (ID A9h)	OpenBSD (ID A6h)	FreeBSD (ID A5h)
a	root (»/«)	root (»/«)	root (»/«)
b	Swap-Bereich	Swap-Bereich	Swap-Bereich
c	complete label	complete label	complete label
d	complete disk	wird zuletzt vergeben	frei verwendbar
e	bel., meist »/usr«	bel., meist »/usr«	bel., meist »/usr«
f	beliebig	beliebig	beliebig
g	beliebig	beliebig	beliebig
h	beliebig	beliebig	beliebig
i-p	ab 1.6 verfügbar	Ab 2.X verfügbar	bis 4.x nicht verfügbar

Tabelle 2.2: Zuordnung der Partitionsbuchstaben bei BSD auf i386

Falls Sie mehrere Betriebssysteme auf dem Rechner installiert haben, merkt das das Installprogramm von NetBSD und zeigt Ihnen nach dem Abschluß von *fdisk* das Installationsmenü des Bootumschaltprogramms. Dieses Buch enthält in Kapitel 3.4 eine ausführliche Beschreibung zum Bootswitcher von NetBSD und zu seinen zahlreichen Optionen. An dieser Stelle soll deshalb nur die Vorgehensweise zur Installation kurz beschrieben werden.

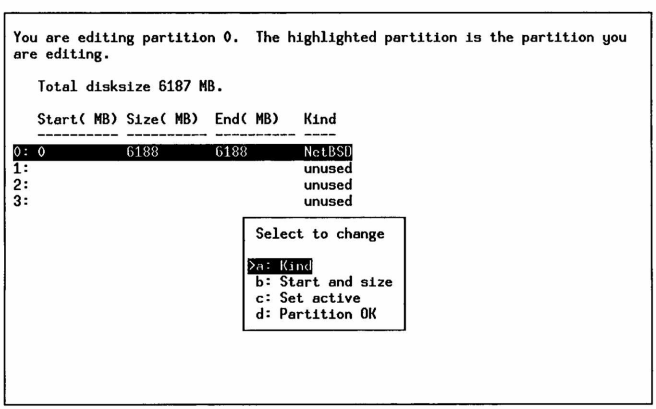


Abb. 2.10: Die Optionen für den ersten Slice. In diesem Fall ist bereits eine NetBSD-Partition eingerichtet. Die anderen Partitionen sind unbenutzt

Grundsätzlich handelt es sich beim NetBSD-Bootselektor um einen Master Boot Record, mit dem nur dynamisch das Active-Flag (siehe Kapitel 3.3) gesetzt wird. Der

Bootswitcher erlaubt nur die Umschaltung primärer Partitionen und (**wichtig!**) überschreibt auch einen im Master Boot Record installierten LILO so gründlich, daß ein eventuell installiertes Liux nicht mehr startfähig ist. Sie sollten ihn also nur einsetzen, wenn:

- NetBSD zusammen mit Windows oder anderen BSDs eingesetzt wird.
- Mehrere NetBSD-Varianten sich auf verschiedenen Festplatten befinden.
- Kein Linux mit LILO, GRUB oder einem anderen linuxspezifischen Bootswitcher verwendet wird.

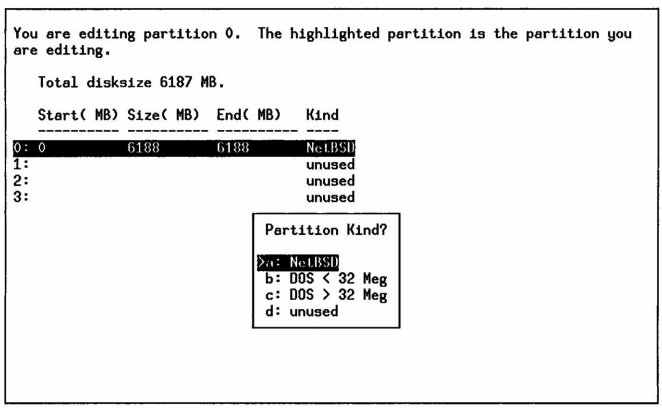


Abb. 2.11: Die Auswahl des Partitionstyps für den gewählten Slice. Die Auswahl ist minimal, es können nur NetBSD-Partitionen (ID A9h) und primäre DOS-Partitionen angelegt werden

Das Installprogramm besitzt einige sehr einfach zu verstehende Optionen. Wie bei *fdisk* auch wird ein kleines Menü für jede vorhandene Partition angeboten. Beachten Sie, daß nur die Einträge zur korrekten Änderung der Partitionen sichtbar sind. Zusätzlich können die Wartezeit bis zum Booten des voreingestellten Betriebssystems (*Set timeout value*) und das voreingestellt zu startende System angegeben werden (*Set default option*). Der Hauptbildschirm zeigt alle von NetBSD erkannten Partitionen und bietet sie als Bootеintrag an. Bei Notebooks können dies durchaus auch Auslagerungspartitionen sein, deren Angabe im Menü sinnlos ist. Grundsätzlich sollten Einträge mit Kennungen wie *Unknown* nicht verwendet werden. Die Liste der von NetBSD erkannten Betriebssysteme ist groß und es ist unwahrscheinlich, daß es sich bei solchen Partitionen um sinnvolle Wahlmöglichkeiten handelt.

Bevor im Disklabel-Editor die Einteilung des NetBSD-Slice erfolgt, will *sysinst* die ungefähre Größe der Distribution wissen (Abb. 2.12). Das macht heute zwar wenig Sinn, da auf modernen Rechnern mit Plattenplatz nicht gespart werden muß, auf älteren Systemen ist die Voreinstellung aber recht hilfreich.

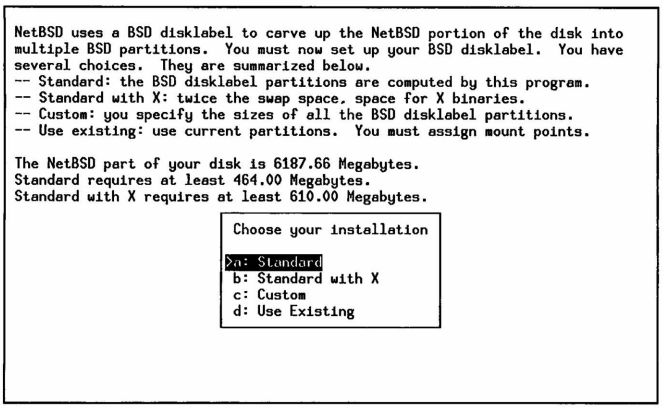


Abb. 2.12: Die Auswahl der Grundinstallationsart

a: Standard ist die Installation des Grundsystems. Es wird keine grafische Oberfläche installiert. Verfügbar sind der C-Compiler, der Perl-Interpreter und weitere Kommandozeilentools.

b: Standard with X enthält das Standard-Paket und zusätzlich die grafische Oberfläche. Außerdem wird die Größe des Auslagerungsbereichs verdoppelt, um mit X flüssig arbeiten zu können. Man sollte sich von der X-Installation aber nicht allzuviel erwarten. Als einziger Window Manager ist *twm* enthalten. Andere Window Manager muß man über das Package-System installieren.

Das System ist so intelligent, daß es die zu entpackenden Daten und die Partitionsgrößen nach dem verfügbaren Plattenplatz kalkuliert und auch die Größe des zu installierenden Systems anzeigt.

Bei *c: Custom* werden keine Voreinstellungen kalkuliert. Man hat die Wahl, was denn nun installiert werden soll. Dies ist nur für erfahrene Anwender, die schon mehrfach NetBSD installiert haben, zu empfehlen.

Standardmäßig sind alle Installationssets ausgewählt. Wenn man an Platzmangel leidet oder ein schlankes System will, braucht nur der »Pflichtteil«, der die ersten drei Pakete umfaßt, installiert werden. Ohne diese Sets ist NetBSD nicht lauffähig.

Im Normalfall reicht aber die Anwahl von Punkt »a« oder »b«. Das System kalkuliert nun die Größen und bietet eine Vorauswahl bei der Partitionierung an.

Grundsätzlich werden mindestens zwei Partitionen *a* für das Root-Dateisystem und *b* für den Auslagerungsbereich benötigt. Der Auslagerungsbereich sollte im Normalfall immer doppelt so groß gewählt werden wie der im Rechner vorhandene physikalische Arbeitsspeicher. Allerdings schadet es auch nicht, wenn erheblich mehr Swap-Space konfiguriert wird, kritisch ist es nur, wenn zu wenig Auslagerungsplatz vorhanden ist. Stellt man so etwas nachträglich fest, hilft nur noch das Anlegen einer zusätzlichen Auslagerungsdatei (siehe Kapitel 3.3).

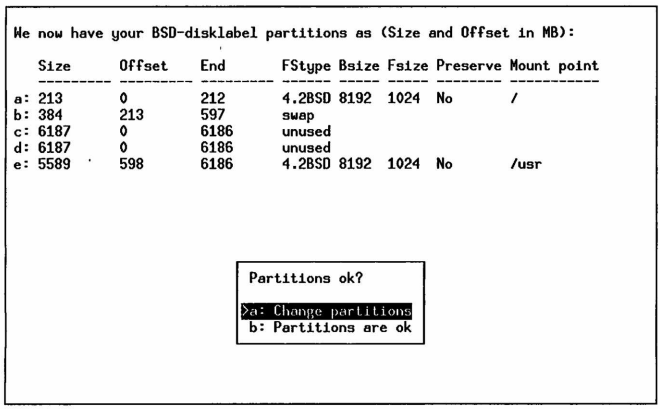


Abb. 2.13: Auswahl der Partitionsgrößen und der Mountpunkte

NetBSD kommt im übrigen auch blendend mit Auslagerungspartitionen von FreeBSD zurecht. Sollte durch eine Parallelinstallation von FreeBSD auf einer anderen Festplatte dort eine Swap-Partition angelegt sein, kann sie ebenfalls eingebunden werden.

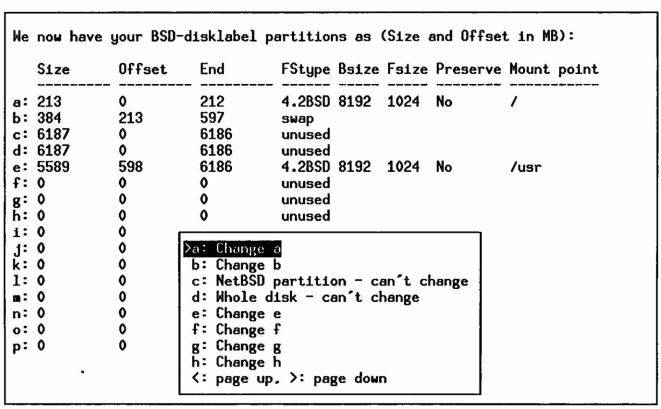


Abb. 2.14: Die möglichen einzurichtenden Partitionen

Sind die Partitionen angelegt und Mountpunkte zugewiesen, wird der Disklabel-Editor durch Anwahl des Menüpunkts *b: Partitions are ok* (Abb. 2.13) verlassen. Es wird nun in einem weiteren Bildschirm nach dem Namen für das Disklabel gefragt (Abb. 2.15). Es kann die Voreinstellung *mydisk* belassen werden. Der Name hat für die Funktion selbst keine Bedeutung.

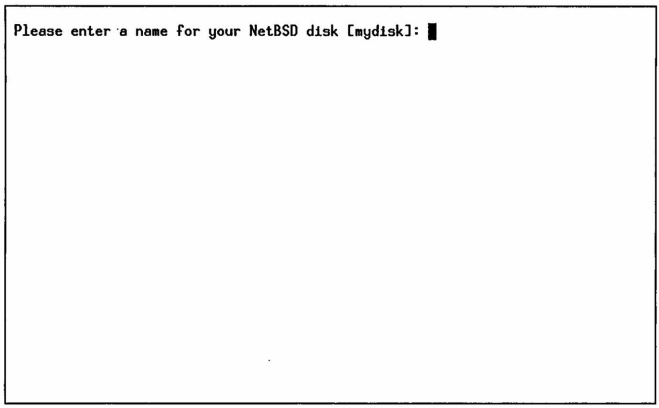


Abb. 2.15: Die Vergabe des Namens für das Disklabel. Der Name selbst spielt keine Rolle

Wird hier mit [Enter] bestätigt, folgt eine letzte Abfrage, ob tatsächlich die in den vorherigen Dialogen getroffenen Einstellung korrekt waren und ob fortgefahren werden soll (Abb. 2.16). Erst wenn dieser Dialog bejaht wird, werden Daten auf die Festplatte geschrieben.

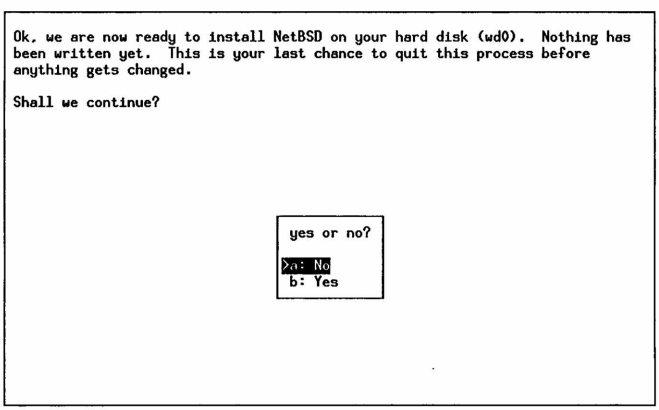


Abb. 2.16: Jetzt wird es ernst!

Der schwierigste Teil der Installation ist nun geschafft.

Nachdem die Festplatte für das Betriebssystem vorbereitet ist, müssen die Installationspakete gewählt werden. Zur Auswahl stehen die Standardinstallation, eine Standardinstallation mit X Window und eine angepaßte Installation.

fips

In der vorhergegangen Beschreibungen wurde davon ausgegangen, daß eine Partition ausreichender Größe für NetBSD zur Verfügung steht.

Anders sieht es aus, wenn Sie einen Rechner haben, der noch in Betrieb ist und von dessen Platte etwas Platz für NetBSD abgezweigt werden soll. Gesetzt dem Fall, daß diese Platte hinreichend groß ist, ist es auch möglich, eine DOS- oder FAT-Partition nachträglich zu verkleinern. Wenn es sich dabei um eine primäre Partition im DOS-Format handelt, kann das mit einem kleinen, feinen Freeware-Tool namens *fips* getan werden.

Starten Sie unter Windows *fdisk* und lassen Sie sich alle Partitionen anzeigen. Wenn Sie feststellen, daß alle Laufwerke auf der Platte primäre Partitionen sind (normalerweise ist das genau eine), können Sie sich freuen: Es wird recht einfach werden, Platz für NetBSD zu schaffen, sofern diese Partitionen noch hinreichend Platz bieten. Falls sich die anderen Laufwerke auf erweiterten Partitionen befinden, ist eine umfangreiche Datensicherung und das Löschen aller Laufwerke auf der Partition und schließlich das Löschen der Partition selbst nötig. Danach muß eine kleinere DOS-Partition angelegt werden und ein hinreichend großer Rest kann für NetBSD bleiben.

Gehen wir von dem Fall aus, daß alle Partitionen auf der Platte primär sind. Führen Sie ein *Scandisk* aus, um sicherzustellen, daß das Laufwerk einwandfrei ist. Anschließend wird das entsprechende Laufwerk defragmentiert, um sämtliche Daten auf die ersten Zylinder, also an den Anfang der Partition, zu bekommen. Dann finden Sie heraus, wieviel Platz noch ist. Führen Sie zur Sicherheit unbedingt eine Datensicherung durch. Starten Sie *fips* in einem MS-DOS-Fenster. Befolgen Sie die Anweisungen des Programms, um die Prozedur durchzuführen.

Auch das, was *fips* kann und was nicht, bedarf der Erklärung: *fips* bearbeitet nur DOS-Partitionen. Mit VFAT- oder NTFS-Partitionen funktioniert es nicht. Hier ist man auf kommerzielle Tools wie *Partition Magic* angewiesen.

Im Normalfall sollte man hier *Custom* wählen, da dort die beste Auswahlmöglichkeit besteht. Sie gelangen auf einen Bildschirm, der eine Auswahl der Pakete zeigt, die auf der CD zur Verfügung stehen. Schauen Sie sich um. Von *sysinst* ist hier vorgegeben, daß alles installiert wird. Sie können die Pakete, die Sie nicht wollen, abwählen. Ein Kandidat dafür ist das Paket »games«. Es handelt sich um recht alte Textmodus-Spiele, die heute niemand mehr aufruft. Ansonsten können Sie aber auch alles angewählt lassen. Dieser Zustand entspricht dann *Standard with X*. Für ein Desktop-System ist das keine falsche Entscheidung.

Wenn der Platz knapp ist oder der Rechner als Router oder Server verwendet werden soll, kann aber auf viele Pakete – insbesondere die ganzen X-Dateien – verzichtet werden. Um NetBSD überhaupt lauffähig zu bekommen, werden nur die Pakete *kernel*, *base* und *system* benötigt. In diesem Fall kommt das System mit etwa 80 Mbyte plus Speicherplatz für die tägliche Arbeit und die Logdateien aus. Es werden dabei der GENERIC-Kernel, das Grundsystem mit allen »lebensnotwendigen« Tools und die

Konfigurationsdateien installiert. Entwicklungstools wie der C-Compiler und auch die Headerdateien kommen dabei nicht auf die Maschine. Es ist damit also nicht möglich, einen neuen Kernel zu bauen oder das Package-Source-System zu benutzen.

Wollen Sie nachträglich hier abgewählte Pakete einspielen, geht das sehr einfach. Sie benötigen dazu nur die Original-CD, diese müssen Sie normal mounten und dann ist aus dem Hauptverzeichnis des NetBSD-Systems der Befehl

```
tar -xvzf /Mountpunkt_zur_CD/i386/binary/sets/Paketdatei.tgz
```

aufzurufen.

Panne!

Wenn etwas dazwischen kommt, das heißt, daß das Programm das Laufwerk nicht erkennt oder es nicht findet, können Sie sysinst auch verlassen und nachprüfen, wo die CD zu finden ist.

Drücken Sie [Ctrl]-[Z], um das Installationsprogramm zu unterbrechen.

Sie gelangen dann in eine Shell. Denken Sie daran: Das System glaubt immer noch, es sei in Kalifornien. Seien Sie mit der Tastatur entsprechend umsichtig. Die wohnt aus Systemsicht nämlich auch in den USA.

Einen Slash (»/«) finden Sie auf einer deutschen Tastatur auf dem Zeichen für den Bindestrich.

Mit *cat /kern/msgbuf* können Sie die Kernelmeldungen des Bootvorgangs lesen. Wenn das Scrollen zu schnell geht, bietet sich auch ein Editor an, auf den Sie bereits zugreifen können.

Geben Sie *ed /kern/msgbuf* ein.

Suchen Sie nach Informationen zu Ihrem CD-ROM-Laufwerk und notieren Sie sich den Gerätenamen.

Beenden Sie den Editor und gehen Sie mit *fg* zurück nach *sysinst*.

Wechseln Sie das Gerät, indem Sie im Menü *Change* auf den Punkt *Device* gehen und geben Sie die ermittelten Daten ein. Wenn sie damit fertig sind, gehen Sie auf *continue*.

Wird das Auswahlmenü über Anwahl des Punktes *x: exit* verlassen, gelangt man zur Abfrage des Installationsmediums. Zur Auswahl stehen NFS, ftp, CD-ROM, das Laden von einem nicht eingehängten Verzeichnis, über Disketten, oder einem bereits gemounteten Verzeichnis heraus. Die einfachste Installationsart geschieht über eine CD-ROM. Man muß nur den Datenträger einlegen (normalerweise sollte er beim Booten von CD-ROM ohnehin eingelegt sein) und bestätigen. Es erscheint wieder ein Auswahlbildschirm, der einige Optionen anbietet. Hier haben Sie die Möglichkeit, das Gerät, von

dem installiert werden soll, auszuwählen. Meistens wird das das erste CD-Laufwerk, das unter NetBSD die Kennung »cd0« hat, sein. Gehen Sie auf den Punkt *cdrom* und es erscheint der nächste Bildschirm, auf dem Sie gefragt werden, ob *sysinst* Ihnen mitteilen soll, welche Dateien installiert werden sollen. Das bleibt Ihnen überlassen. Bestätigen Sie das ruhig, um mitzubekommen, ob das System nicht doch aus irgendeinem Grunde hängenbleibt. Jetzt werden die gewünschten Daten von der CD auf das System kopiert.

2.1.2 Die Installation über das Netzwerk

Eine Installation via Netzwerk ist immer dann interessant, wenn Sie entweder kein Geld für ein CD-ROM-Laufwerk ausgeben wollen oder der Rechner so alt ist, daß das BIOS nicht in der Lage ist, ein CD-Laufwerk zu erkennen, wie es bei sehr alten Maschinen der i386-Kategorie manchmal der Fall ist, und/oder Ihr altes Laufwerk defekt ist. Es mag aber auch vorkommen, daß wir Sie jetzt zu einer neuen »Schandtat« angeregt haben. Versuchen sollten Sie es auf jeden Fall einmal.

Die Nachteile dieses Verfahrens sind, daß der Wirtsrechner auch ein Unix-Verwandter (zum Beispiel Linux oder HP-UX) ist. Außerdem muß der »Neue« auf einen lokalen Nameserver zugreifen können und in diesen eingetragen sein. In so einem Fall reicht es, wenn Sie sich das Verzeichnis »sets« vom FTP-Server der NetBSD-Organisation heruntergeladen haben. Sie finden es unter *ftp://ftp.netbsd.org/pub/NetBSD/1.6/i386/binary/*.

Legen Sie es unter einem Verzeichnis auf Ihrem NFS-Server ab und geben Sie das Verzeichnis zum Export frei. Merken oder notieren Sie sich den Pfad.

Starten Sie das System mit dem Installationsprogramm von den Disketten und führen Sie alle Schritte bis zum Formatieren der Festplatte wie bei der Standardinstallation aus.

Dann werden sie wieder gefragt, welches Installationsmedium Sie verwenden wollen. Diesmal wählen Sie »NFS«.

Sie gelangen in einen Bildschirm, der Sie nach den folgenden Informationen fragt: Rechnername, IP-Adresse des Nameservers, Name des Rechners, in dem sich die CD befindet und unter welchem Verzeichnis auf diesem Rechner das »sets«-Verzeichnis gemountet ist. Sie werden gefragt, ob Sie eine automatische IPv6-Konfiguration möchten. Verzichten Sie bitte darauf, bis Sie besser eingearbeitet sind.

Mit diesen Daten füttern Sie *sysinst*.

Wenn das getan ist und alles funktioniert, werden die Dateien nun aus dem Laufwerk auf der anderen Maschine auf den Neuen kopiert. Als Bonbon können Sie die Rechnerdaten gleich in die Systemkonfiguration übernehmen und sich einen Arbeitsschritt sparen.

2.1.3 Die Installation per FTP

Eine Installation per FTP ist ebenfalls möglich, wenn Sie arm an Platz sind oder das Ganze nur mal ausprobieren wollen. Sie wählen einfach ein anderes Installationsmedium, nämlich »ftp«, und beantworten die Fragen von *sysinst* nach Rechnernamen, Gateway und so weiter. Dann will das Installationsprogramm wissen, von wo es sich die Daten besorgen soll. Das ist normalerweise der URL *ftp://ftp.netbsd.org/pub/NetBSD/1.6/i386/binary/*. Anschließend passiert dasselbe wie bei der NFS-Installation auch: die gewünschten Pakete werden auf dem Rechner installiert.

2.1.4 Die Grundkonfiguration

Mit dem Einspielen der Pakete ist es noch nicht getan. *sysinst* fragt noch einige weitere Daten ab. Sie finden diese Abfragen auch in getrennten Menüs (Utilities und andere) von *sysinst* (Abb. 2.17). Es macht aber wenig Sinn, diese Utilities vor dem Abschluß der anderen Arbeiten zu starten. Beispielsweise ist die Auswahl der Zeitzone anfangs leer (Abb. 2.18).

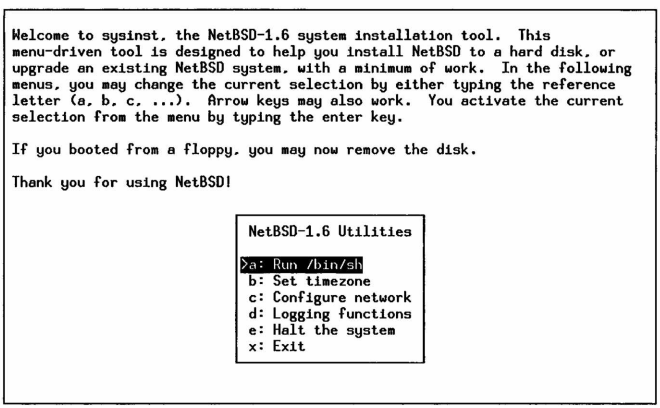


Abb. 2.17: Das Utilities-Menü von sysinst

Wählen Sie also die Zeitzone aus, in der der Rechner betrieben wird. In Deutschland ist das Europe/Berlin, in der Schweiz Europe/Zurich und in Österreich Europe/Vienna. Bei der Abfrage der Passwortverschlüsselung ist normalerweise die ausgewählte Voreinstellung DES sinnvoll.

Außer der Einstellung der Zeitzone ist im Utility-Menü aber besonders die Vorkonfiguration der Netzwerkkarte interessant. Die Konfiguration läuft in einigen wenigen Schritten ab und zeigt auch, ob der Installationskernel überhaupt ein brauchbare Netzwerkkarte gefunden hat (Abb. 2.19). Wenn ein DHCP-Server im Netz verfügbar ist, kann auch gleich getestet werden, ob alles klappt. Es muß dazu nur im nächsten Menü mit »Yes« bestätigt werden (Abb. 2.20).

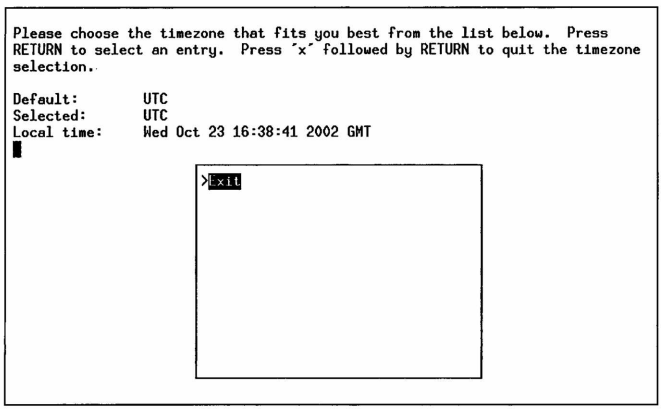


Abb. 2.18: Anfangs läßt sich hier noch gar nichts wählen

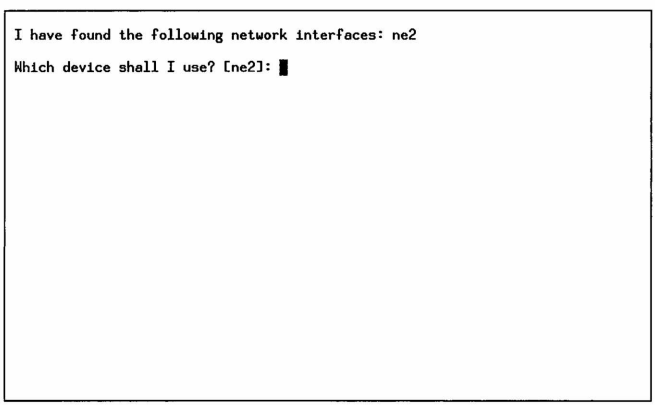


Abb. 2.19: Es wurde eine Netzwerkkarte zur Konfiguration gefunden

Ist kein DHCP-Server vorhanden, muß die Netzwerkkarte noch manuell konfiguriert werden. Eventuell vorher schon gefundene Werte (beispielsweise aus einem erfolgreichen Versuch, DHCP zu starten) werden auch gleich angezeigt (Abb. 2.21).

Sie sollten sich aber von den hier getroffenen Einstellungen nicht zu viel erwarten. Während bei diversen NetBSD-Installationen die Werte ordentlich ins System übernommen wurden, versagte *sysinst* bei anderen Installationen an dieser Stelle. Es ist auf jeden Fall angebracht, später die Einstellungen zu prüfen. Sind Sie mit allen getroffenen Einstellungen zufrieden, starten Sie das System neu. Auch das können Sie direkt in *sysinst* durchführen. Je nachdem, ob Sie die Installation von Diskette oder CD-ROM gebootet haben, müssen Sie das Startmedium dann noch entfernen. Außerdem schadet es nicht, beim Neustart die Bootreihenfolge wieder auf die Originalwerte zu setzen.

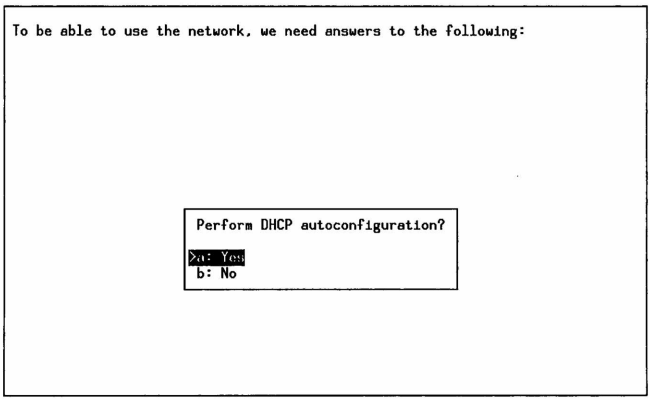


Abb. 2.20: DHCP ist eine feine Sache und spart viel Arbeit

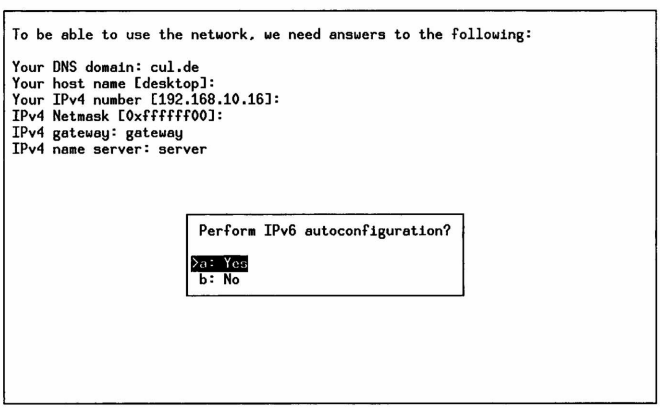


Abb. 2.21: Die manuelle Konfiguration des Netzwerks im Installationsprogramm

sysinst sichern

Das Installationsprogramm *sysinst* ist ausgesprochen praktisch. Man könnte es auch anstelle der regulären Programme *fdisk* und *disklabel* verwenden, wenn es nur mit auf die Platte installiert werden würde. Leider kommt man aber ohne Verrenkungen nach dem ersten Booten nicht mehr an das Programm heran. Aus diesem Grund sollte man vor dem Neustart das Programm von der Diskette oder vom Diskettenimage der CD-ROM auf die Festplatte kopieren. Am besten ist es im Verzeichnis »/stand« der NetBSD-Installation aufgehoben. Falls alle Stricke reißen, kann man sich das Programm auch aus dem NetBSD-Quellen neu kompilieren. Das ist aber sehr viel aufwendiger und wird an anderer Stelle dieses Buchs beschrieben.

2.1.5 Das Booten des installierten NetBSD-Systems

Ist NetBSD erst einmal installiert, sollte es auch gestartet werden können. Es gibt dafür mehrere Möglichkeiten. Am einfachsten funktioniert es, wenn nur ein NetBSD auf der Festplatte installiert ist. Dann ist die NetBSD-Partition im Master Boot Record als »aktiv« vermerkt und wird automatisch gestartet. PCs starten in der Voreinstellung immer von der ersten Festplatte die aktiv vermerkte Partition. Hier erscheint der NetBSD-Bootprompt:

```
>> NetBSD/i386 BIOS Boot, Revision 2.12
>> (autobuild@tgm.daemon.org, Sun Sep  8 1922:13 UTC 2002)
>> Memory 639/97280 k
Press retun to boot now, any other key for boot menu
booting hd0a:netbsd - starting in 0
```

Gibt man nichts an, wird nach der voreingestellten Zeit der Kernel »/netbsd« geladen. Will man einen anderen Kernel booten, beispielsweise eine Sicherung, drückt man nur die Leertaste und gibt am erscheinenden Prompt die folgende Anweisung ein:

```
type "?" pr "help" for help.
> boot wd0a:netbsd-GENERIC
```

oder wie der zu startende Kernel heißt. Man hat zu diesem Zeitpunkt eine amerikanische Tastaturbelegung, so daß anstelle des Doppelpunkt die Taste [ö], anstelle des »-«-Zeichens [ß] zu drücken ist. Auf diese Weise startet man auch eine zweite NetBSD-Installation, beispielsweise auf der zweiten Festplatte:

```
boot wd1a:netbsd
```

Ähnlich wie beim Mac68k, auf dem NetBSD aus dem Host-Betriebssystem heraus gebootet werden muß, besteht eine solche Option auch auf dem Intel-PC. Das ist hilfreich, wenn beispielsweise das Booten von Festplatte aus irgendwelchen Gründen nicht funktioniert, sei es, daß die Hardware defekt ist, oder daß ein Bootmanager nicht installiert werden kann. Dann verwendet man das Tool »dosboot.com«, das zur NetBSD-Distribution gehört und sich im Verzeichnis »i386/installation/misc« befindet. »dosboot.com« funktioniert *nicht* unter Windows NT/2000/XP, sondern nur auf echten DOS-basierten Windows-Installationen oder unter DOS selbst. Es wird wie der Bootloader mit den Parametern *Festplattenpartition:Kernel* aufgerufen:

```
dosboot wd0a:netbsd
```

und verdrängt das gerade aktive Betriebsystem.

Woher bekomme ich ein DOS?

Man sollte meinen, daß jeder noch irgendwo eine DOS-Diskette liegen hat und diese bei Bedarf für Grundarbeiten, sei es das Einspielen eines Mini-Betriebssystems oder eines Bootmanagers zur Verfügung hat. Wer keine DOS-Diskette besitzt, muß auf ein DOS-System nicht verzichten. Man erhält es auch kostenfrei. Die erste Adresse ist *www.freedos.org*, wo ein DOS-System nach GNU-Lizenz entwickelt wird, das schon recht fortgeschritten ist. Zahlreiche Experimente mit FreeDOS zeigten aber, daß es an vielen Stellen doch noch Probleme bereitet. Besser ist es, sich eine Version von DR/Novell/Caldera DOS aus dem Internet zu laden. Dieser MS-DOS-Clone funktioniert einwandfrei, hat keine Kompatiblitätsprobleme und liefert auch alle Hilfsprogramme mit.

Die Distribution dieses DOS ist nicht erlaubt, das Herunterladen und Verwenden für private Zwecke aber sehr wohl. Die Download-Adresse von Caldera OpenDOS 7.01 ist *ftp://ftp.lineo.com/pub/drdos/OpenDOS.701*. Neuere Versionen sollten nicht heruntergeladen werden, da hier die Lizenzbedingungen problematisch sind.

NetBSD läßt sich auch mit den Bootmanagern anderer Betriebssysteme einsetzen.

OS/2-Bootmanager

Der OS/2-Bootmanager belegt eine eigene Partition auf der Platte. Er kann nur von OS/2 oder Partition Magic in der Version 2 bis 3 angelegt und verwaltet werden. Je nach Verwaltungstool, also *fdisk* unter OS/2 oder Partition Magic, wird die Partition mit einem beliebigen, auf acht Buchstaben begrenzten, Namen eingetragen. Naturgemäß kennt der OS/2-Bootmanager die Partions-ID A9h von NetBSD nicht und zeigt deshalb in der erweiterten Darstellung nicht den Partitionsstyp, sondern nur die ID an.

Vom OS/2-Bootmanager ist grundsätzlich Abstand zu nehmen, wenn auf der selben Maschine Windows 2000 installiert ist, da dieses Betriebssystem den Bootmanager aus einer Gemeinheit Microsofts heraus grundsätzlich zerstört. Frühere Windows-Versionen zeigen diese Unart nicht.

LILO

Ist auf dem Rechner parallel zu NetBSD ein Linux-System installiert, kann NetBSD in den Linux-Loader eingetragen werden. Dazu muß in der Datei »/etc/lilo.conf« der Eintrag

```
other=/dev/hd<Nummer> <Partitionsnummer>
label=netbsd
table=/dev/hd<Nummer>
```

mindestens um die oberen Zeilen ergänzt werden. Für *Nummer* und *Partitionsnummer* sind die Werte von Linux anzugeben. Die erste Festplatte heißt »hda«, die erste Partition dann folgerichtig »hda0«. Nach der Eintragung in den LILO muß unter Linux noch das

Programm *lilo* selbst aufgerufen werden, damit die Eintragung auch übernommen wird. Sie sollten auch die Manpages zu *lilo* und *lilo.conf* konsultieren.

EasyBoot

FreeBSD besitzt einen Bootloader, der in den Master Boot Record der Festplatte eingetragen wird. Um EasyBoot zu verwenden, muß kein FreeBSD installiert werden. Es gibt auch ein DOS-Programm, mit dem man den Bootloader schreiben kann. Man lädt sich den EasyBoot-Startsektor zusammen mit dem DOS-Installationsprogramm vom FreeBSD-Server, kopiert das Programm und die Easyboot-Datei auf eine DOS-Diskette und startet hier das DOS-Programm.

EasyBoot besitzt gegenüber dem nativen NetBSD-Bootswitcher den Nachteil, daß er nicht so gut zu konfigurieren ist. Sollte er aber bereits installiert sein, kann er durchaus weiterverwendet werden, sofern NetBSD von der ersten oder zweiten Festplatte bootet.

2.2 Die Installation auf dem Mac68k

von Jörg Braun

Das Gros der Maschinen, auf denen ein NetBSD läuft, ist intel-basiert. Aber bei der großen Zahl von Hardware-Plattformen, die dieses Betriebssystem unterstützt, gehören nicht nur »Exoten«, die in Stückzahlen unter Tausend produziert wurden. Eine solche recht weit verbreitete Hardware-Plattform ist der Apple Macintosh. Die Apple-Portierung ist ohnehin eine der ältesten und hatte ursprünglich den Namen »MacBSD«.

Während es heute für die modernen mit PowerPC-Prozessoren ausgerüsteten Macintosh-Rechner mit MacOS X und der Unterlage Darwin ein spezielles BSD-Derivat gibt, ist man bei den alten Maschinen, wenn man ein unix-artiges Betriebssystem aufsetzen will, auf NetBSD, OpenBSD oder Linux angewiesen. An dieser Stelle soll natürlich die Installation von NetBSD beschrieben werden, die von OpenBSD läuft eigentlich genauso ab.

Was braucht man an Hardware? Eigentlich nur einen alten Macintosh mit 68030- oder 68040-Prozessor, einen Monitor, falls dieser nicht eingebaut ist, und entweder ein CD-Laufwerk oder eine NUBus-Netzwerkkarte. Die Liste der unterstützten Maschinen ist recht lang, trotzdem sollte man vor dem Versuch der Installation auf dem NetBSD-Server nachsehen, ob die Hardware wirklich von der aktuellen Betriebssystemversion unterstützt wird. Unter anderem läuft NetBSD bis zur Version 1.6 beispielsweise nicht auf dem Quadra 900.

Im Gegensatz zu anderen Maschinen booten 68k-Macintoshs immer MacOS, egal ob Version 7.x oder 8. Sie booten das Betriebssystem aber von einem beliebigen Gerät, das kann also eine CD-ROM, eine Floppy oder eine Festplatte sein. Im Gegensatz zu neuen Macintosh-Maschinen, die auf preiswerte Massenware wie IDE und USB bei der Hardware setzen, wurde in den alten Rechnern nur feinste Peripherie verarbeitet, als Schnittstelle zu den Massenspeichern dient der SCSI-Bus.

Die hier beschriebene Installation wurde auf einem Quadra 700 mit einer vier Gbyte großen Festplatte, einem externen CD-Laufwerk, 20 Mbyte Hauptspeicher und einer 10 MBit-NUBus-Ethernetkarte von Asanté durchgeführt. Es handelt sich bei dem Rechner also um ein ehemaliges Highend-Gerät. Für heutige Maßstäbe ist der Rechner mit seiner 25 MHz-CPU völlig veraltet, er wird im normalen Betrieb nicht mehr eingesetzt. Als Host-Betriebssystem wird MacOS 7.5.3 verwendet, das vom Apple-Server auch als Diskettenimage-Set bei Bedarf heruntergeladen werden darf[1].

MacOS, ISO 9660, Joliet, HFS und Rockridge

Das normale Format für die Verzeichnis- und Dateinamen von CD-ROMs ist ISO 9600. Damit sind aber nur Dateinamen nach der 8+3-Konvention möglich. Außerdem dürfen die Dateinamen nur Großbuchstaben und dabei die Buchstaben A bis Z und die Zahlen 0 bis 9 enthalten. Verzeichnisse dürfen nur acht Zeichen lang sein und keine Endung besitzen. MacOS kann von Haus aus nur CDs verarbeiten, die entweder im Macintosh-eigenen HFS- oder im ISO-Format gebrannt sind. MacOS kann weder die Joliet-Extensions von Windows, noch das Rockridge-Format von Unix verarbeiten. Unter *http://www.tempel.org/joliet/* und *ftp://ftp.tempel.org/pub/Mac/* kann aber ein Treiber heruntergeladen werden, dessen freie Version 1.4.2 hervorragend mit MacOS 7.5 und 8.0 funktioniert (unter MacOS 8.6 und höher sollte laut Autor der Software besser die neuere Shareware-Version 2.x verwendet werden). Mit dem Einbinden des Freeware-Treibers werden die langen Dateinamen von Joliet-CDs unter MacOS korrekt verarbeitet. Wenn also unter Unix CD-ROMs mit *mkisofs* gemastert werden, die auch auf einem Macintosh vernünftig gelesen werden sollen (was bei den Installations-CDs von NetBSD der Fall ist), sollte der Schalter *-J* auf der Kommandozeile von *mkisofs* auf keinen Fall vergessen werden.

Da der Rechner keinen direkten Internet-Zugang hat, wurden die NetBSD-Pakete vorab mit einer anderen Maschine heruntergeladen und auf einem lokalen anonymen FTP-Server zwischengelagert. Das Archiv mit den Package-Sourcen wurde aus Bequemlichkeit direkt von der CD-ROM zur Ausgabe 6'2002 der Zeitschrift freeX übernommen. Um die langen Dateinamen von der CD-ROM zu erhalten, ist ein Freeware-Joliet-Treiber unter MacOS installiert.

Im Gegensatz zu anderen Installationen wie beispielsweise auf der Intel-Plattform geschieht die Vorkonfiguration und das Einspielen der Software nicht durch Booten mit einer NetBSD-Diskette oder -CD-ROM, sondern vom Hostsystem aus. Man benötigt also für die Installation auch einige MacOS-Programme und einen installierten Stuffit Expander sowie zum Neueinrichten der Festplatte eine Partitionierungssoftware wie beispielsweise dem Ontrack Diskmanager Mac.

---

[1] *ftp://ftp.apple.com/Apple_Support_Area/Apple_Software_Updates/German/Macintosh/System/Full_Installs/System_7.5_Version_7.5.3/*

Partitionsgrößen

Auf dem Macintosh mit 68k-Prozessor ist das Anlegen der Unix-Partitionen ein Geduldsspiel, weil bei zu groß gewählten Partitionen diese zwar angelegt werden, anschließend aber unbrauchbar sind. Gleichzeitig ist es nicht möglich, in der Mini-Konsole die zusätzlichen Partitionen zu einzubinden. Bewährt haben sich Partitionsgrößen knapp unter 500 Mbyte. Damit können bei der Grundinstallation die wichtigen Daten auf die Root-Partition kopiert werden, dann das System in den Grundzustand versetzt und die Datei »/etc/fstab« editiert werden. Anschließend lassen sich von MacOS aus die restlichen Pakete problemlos installieren. Alternativ kann man die zusätzlichen Pakete auch von NetBSD aus mit *tar* entpacken, wenn man einen lokalen NFS- oder FTP-Server laufen hat.

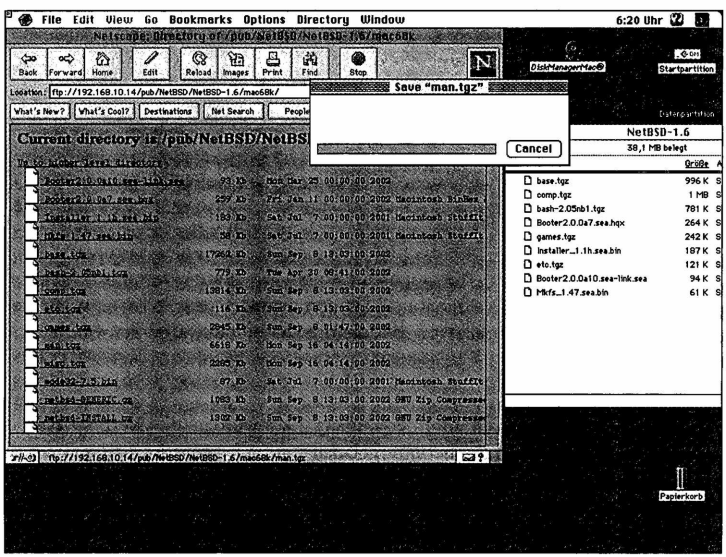


Abb. 2.22: Die Übername der Daten via Netscape-Browser vom lokalen ftp-Server

Eine erste Hürde lauert hier schon vor der eigentlichen Installation. Während es auf dem Intel-PC mit verschiedenen Tools möglich ist, die Plattenkonfiguration – auch die Partitionierung der Boot-Festplatte – im laufenden Betrieb zu ändern, geht das auf dem Macintosh nicht. Es muß also dafür von einem anderen Medium, entweder von einer Diskette oder von einer anderen Festplatte, gebootet werden. Hier wurde letzteres gewählt und dann die lose im System befindliche Festplatte nach der Partitionierung wieder ausgebaut. Der Vorteil an dieser Vorgehensweise ist unter anderem auch, daß man, wenn man einen Fehler macht, nicht gleich das gesamte MacOS ebenfalls neu installieren muß. Man muß nur die Festplatte unter MacOS einrichten, alle Daten umkopieren, dann die Platte für das nächste Booten anmelden, die erste Platte partitionie-

ren, die Daten des Betriebssystems zurückspielen und dann die ursprüngliche Platte wieder als Bootplatte definieren.

Nun erst beginnt die Einrichtung von NetBSD mit den MacOS-Hilfsprogrammen, die wie die Installationspakete vom NetBSD-Server heruntergeladen werden.

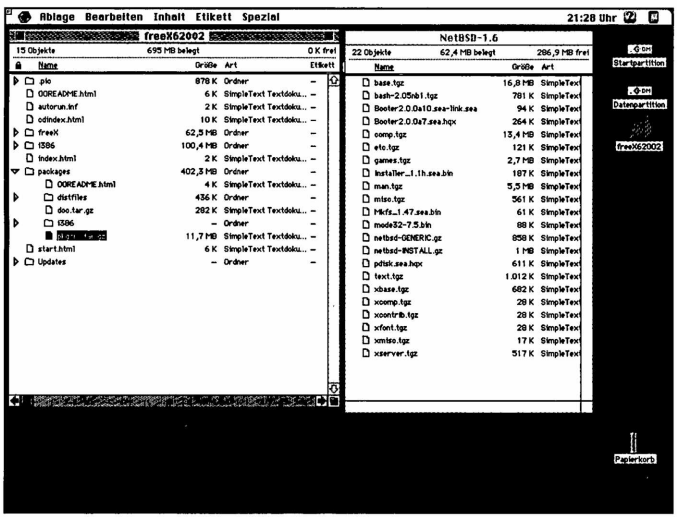


Abb. 2.23: Die Übernahme der Daten von einer CD-ROM gestaltet sich naturgemäß einfacher

Im Gegensatz zur Intel-Plattform verwendet BSD auf dem Macintosh nicht nur einen einzelnen Slice. Vielmehr werden im Regelfall mindestens drei Partitionen angelegt. Als Partitionstypen sind *Apple_UNIX_SRV2 Root*, *Apple_UNIX_SRV2 Swap* und *Apple_UNIX_SRV2 Usr* zu wählen. Das Anlegen erfolgt mit dem Diskmanager Mac oder alternativ mit dem vom NetBSD-Server beziehbaren MacOS-Programm *pdisk 68k* (Abb. 2.25). In diesem Programm werden die Partitionen zugeordnet. BSD-typisch werden die Partitionen aber auch in ein Disklabel aufgenommen, sie werden also anschließend über die selben Kennungen (*sd0a*, *sd0b* und so weiter) angesprochen wie SCSI-Devices beispielsweise bei der Intel-Plattform.

Der Mac-Installer

Der Installer für den Macintosh dient nicht ausschließlich der Installation der Sets, er bietet über das Menü auch eine Mini-Konsole und außerdem einen Menüpunkt zum Anlegen der Gerätedateien in »/dev«. Dieser Menüpunkt ist wichtig und sollte vor der Installation, spätestens aber vor dem ersten Boot-Versuch, unbedingt ausgeführt werden.

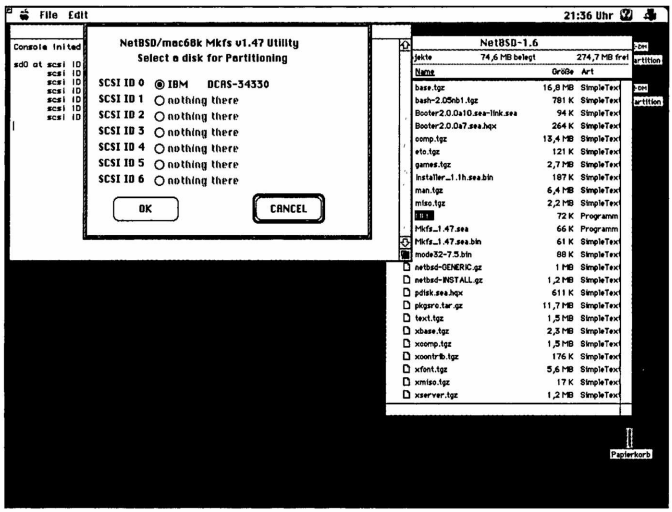


Abb. 2.24: Das Formatierprogramm Mkfs und die Auswahl der Festplatte

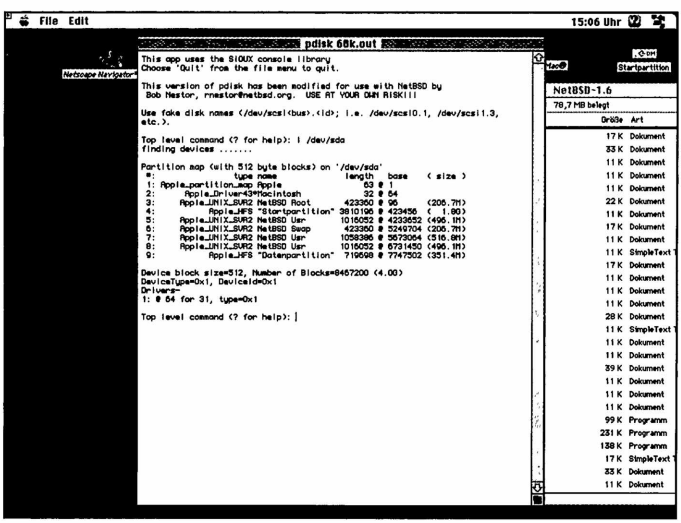


Abb. 2.25: Die Aufteilung der Platte kann statt mit einem kommerziellen Programm auch mit pdisk für m68k vorgenommen werden

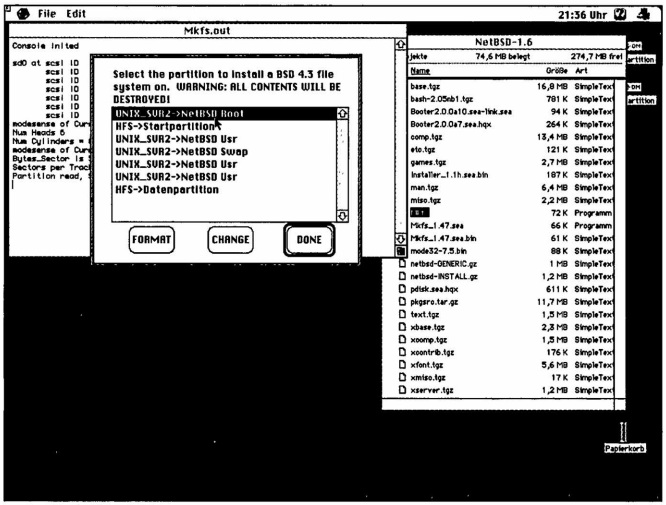


Abb. 2.26: Die Auswahl zu formatierender Partitionen in Mkfs

Nach der Aufteilung der Platte müssen die BSD-Partitionen formatiert werden. Dafür verwendet man Mkfs (Abb. 2.26), das ebenfalls vom NetBSD-Server bezogen werden kann. Sind alle Partitionen (außer der Auslagerungspartition) formatiert, wird im nächsten Schritt der *NetBSD/Mac68k Installer* gestartet. Es handelt sich um eine kleine Konsole, in der erste Tests mit dem neuangelegten NetBSD durchgeführt werden können. Der Installer dient prinzipiell aber dazu, die Software-Sets einzuspielen. Dazu müssen diese Pakete lokal vorhanden sein, die Installation direkt beispielsweise von einem ftp-Server ist nicht möglich. Das Einspielen dauert auf der veralteten CPU recht lange, insbesondere wenn man alle Sets ausgewählt hat. Man sollte sich also in Geduld üben.

Bis zur Version 1.5 wurde die Installation allein über den Installer abgewickelt. Man kann diese manuelle Installation auch bei der Version 1.6 noch durchführen. Alternativ ist auch ein *sysinst*-basiertes Einspielen der Sets möglich. Diese Installation entspricht weitgehend der Installation des Intel-Ports.

Wer schon frühere Versionen von NetBSD auf dem Mac68k kennt, wird bei der Version 1.6 von NetBSD bei der gewohnten Installation eine unangenehme Überraschung erleben. Bei der Version 1.5.2 war der Kernel noch als .tgz-Datei auf dem Server abgelegt. Bei der Version 1.6 handelt es sich um eine direkt gepackte Datei mit der Endung ».gz«. Mit solchen Dateien kann der Installer aber nichts anfangen. Die beste Lösung ist, auf einem anderen Rechner (BSD oder Linux) die Datei aus- und dann wieder richtig einzupacken:

```
gunzip netbsd-GENERIC.gz
tar -xzf netbsd-GENERIC.tgz netbsd-GENERIC
```

Nun kann diese Datei auf den Macintosh übertragen und mit dem Installer eingespielt werden.

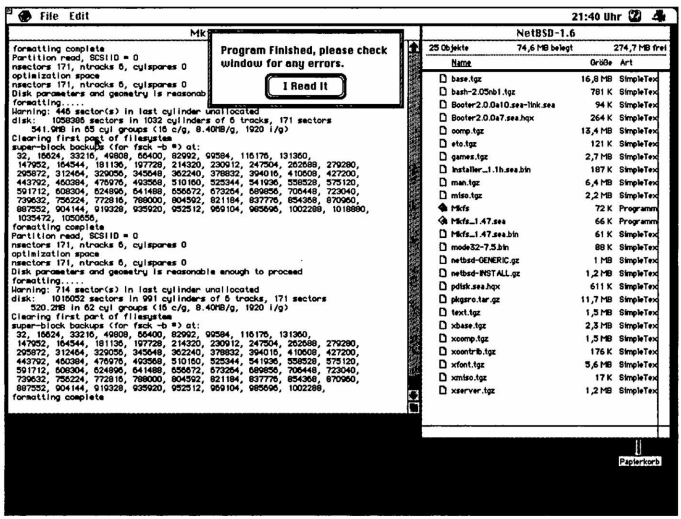


Abb. 2.27: Die Partitionen sind formatiert

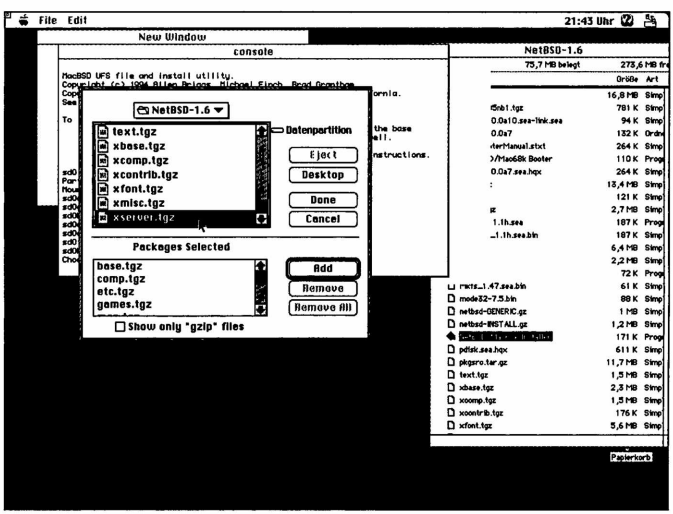


Abb. 2.28: Die Auswahl der Installationssets im Installer. Eine Mehrfachauswahl ist möglich, man sollte der Bootpartition aber nicht zu viel zumuten

Die Mini-Shell besitzt aber noch andere Schwächen. Es gibt zwar einen *mount*-Befehl, der Versuch, ihn einzusetzen, führt aber nur zu einer Fehlermeldung:

```
> mount /dev/sd0g /usr/X11R6
Error on SCSIRead(), # 5
/usr/X11R6 : Invalid argument
>
```

Damit ist die Vorgehensweise klar: Zuerst wird die Platte eingerichtet. Dabei wird eine Root-Partition eingerichtet, die auf jeden Fall wenigstens das Basis-Paket und den Kernel aufnehmen kann. Ausreichend hat sich eine Partitionsgröße von etwa 200 Mbyte erwiesen. Für die Mountpunkte »/usr/X11R6«, »/usr/pkg« und »/usr/pkgsrc« werden ausreichend große Partitionen (jeweils etwa um die 500 Mbyte) angelegt. Außerdem darf der Auslagerungsspeicher nicht vergessen werden. Hier wurde er auf der 4 Gbyte-Platte mit zirka 200 Mbyte äußerst großzügig bemessen.

Paket	Funktion	Größe
base.tgz	Das unumgängliche Basispaket	16, 8 Mbyte
comp.tgz	Compiler und Tools	13,4 Mbyte
etc.tgz	Konfigurationsdateien	121 Kbyte
games.tgz	BSD-Konsolenspiele	2,7 Mbyte
man.tgz	Man-Pages und Info-Dateien	6,4 Mbyte
misc.tgz	Diverse Programme	2,2 Mbyte
text.tgz	Textbearbeitungs-Werkzeuge	1,5 Mbyte
xbase.tgz	X-Basispaket	2,3 Mbyte
xcomp.tgz	X-Header-Dateien	1,5 Mbyte
xcontrib.tgz	X-Zusatzdateien	176 Kbyte
xfont.tgz	X-Schriften	5,6 Mbyte
xserver.tgz	Der X-Server	1,2 Mbyte
netbsd-GENERIC.tgz	Der NetBSD-Kernel. Unabdingbar!	1 Mbyte
pkgsrc.tgz/.tar.gz	Der Package-Source-Tree	ca. 12 Mbyte

Tabelle 2.3: Die Sets für Mac68k bei NetBSD 1.6

Das Einspielen des Betriebssystems mit *sysinst* funktioniert ähnlich wie die manuelle Installation. Man benötigt dazu aber die zusätzliche Kerneldatei »netbsd-INSTALL.gz«. Diese Kernel-Datei enthält nämlich den Installer.

Im Optionen-Menü des NetBSD-Booters wählt man *Boot Options*, wodurch eine weitere Menüleiste aufklappt. Setzen Sie die Checkbox *GMT Bias* unten links und wählen Sie dann die Option *Boot from Mac OS* am oberen Rand des Fensters. Nun ist der rechte Einstellknopf aktiv. Wählen Sie mit ihm die richtige Datei für die Installation des Kernels auf Ihrer Hardware. Dies ist entweder »netbsd-INSTALL.gz« oder »netbsd-INSTALLSBC.gz«. Mit dem Schließen-Knopf schließen Sie das Fenster. Im Gegensatz zur manuellen Installation wird hier also ein Kernel aus dem MacOS-Dateisystem gestartet. Im Optionen-Menü wählen Sie *Boot Now*. Jetzt wird der Installationskernel mitsamt dem Programm *sysinst* gestartet. Diese Installation entspricht praktisch zu hundert Prozent der der anderen Plattformen.

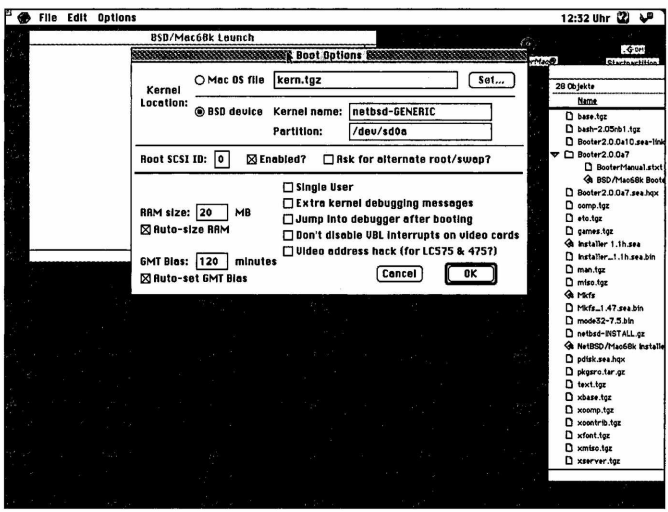


Abb. 2.29: Die Grundoptionen für das Booten von NetBSD aus MacOS heraus

Es wird das Hauptmenü angezeigt, von dem aus die Installation begonnen wird:

```
.*********************************************.
* NetBSD-1.6 Install System                   *
*                                             *
*>a: Install NetBSD to hard disk              *
* b: Upgrade NetBSD on a hard disk            *
* c: Re-install sets or install additional sets *
* d: Reboot the computer                      *
* e: Utility menu                             *
* x: Exit Install System                      *
.*********************************************.
```

Sind der Kernel, das Basis-Paket und bei Bedarf auch noch kleinere Sets eingespielt, wird NetBSD zum ersten Mal mit dem Programm *BSD/Mac68k Booter* gestartet. Beim ersten Start des Programms müssen zuvor die Angaben über Bootplatte und Bootkernel korrigiert werden. Im vorliegenden Beispiel wurde NetBSD auf der Platte mit der SCSI-ID 0 installiert, im Dialog sind deshalb *sd0a* und die SCSI-ID 0 im Dialog einzutragen, außerdem als Boot-Kernel *netbsd-GENERIC*.

Nun kann der Menüpunkt *Boot Now* ausgewählt werden. Das System wird in den Single-User-Mode fallen. Wichtig ist zu beachten, daß während des Bootens von NetBSD die Maus grundsätzlich nicht bewegt werden darf, sonst wird die Tastatur blockiert!

Es kann durchaus passieren, daß der Bootvorgang abbricht oder einfach nichts zu sehen ist. Dann stimmen die Bildschirmeinstellungen nicht. Während NetBSD vor der Version 1.6 nicht in der Lage war zu starten, wenn eine andere Bildschirmeinstellung als 640x480 in ein Bit Farbtiefe eingestellt war, funktionieren jetzt auch andere Auflösungen und Farbtiefen. Bei mir wollte das System bei einer Farbtiefe von einem Bit aber anfangs nicht booten. Mit der Einstellung von 256 Farben klappte es dann. Die Bildschirmeinstellungen sind also mit Vorsicht zu genießen. Probieren Sie es gegebenenfalls aus. Im schlimmsten Fall, wenn Sie gar nichts mehr sehen, müssen Sie den Rechner mit dem Reset-Taster neu starten. Das ist zwar nicht schön, kann aber zu diesem Zeitpunkt keinen Schaden anrichten. Wenn mit den vorgenommenen Einstellungen das System erfolgreich bootete, speichern Sie die Einstellungen vor dem nächsten Start mit *File | Save Options*.

```
[ using 427368 bytes of netbsd ELF symbol table ]
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All rights reserved.

NetBSD 1.6 (GENERIC) #0: Sun Sep  8 20:58:17 UTC 2002
    autobuild@tgm.daemon.org:/autobuild/mac68k/OBJ/autobuild/src/sys/arch/mac68k/
compile/GENERIC
Apple Macintosh Quadra 700  (68040)
cpu: delay factor 800
total memory = 20480 KB
avail memory = 15076 KB
using 281 buffers containing 1124 KB of memory
mrg: 'Mac IIci/Q700 ROMs' ROM glue, tracing off, debug off, silent traps
mainbus0 (root)
obio0 at mainbus0
esp0 at obio0 (quick): address 0x88f000: NCR53C96, 25MHz, SCSI ID 7
scsibus0 at esp0: 8 targets, 8 luns per target
adb0 at obio0
asc0 at obio0: Apple Sound Chip
intvid0 at obio0 0 f9000e00: DAFB video subsystem, monitor sense 1
```

```
intvid0: 832 x 624, 256 color
macfb0 at intvid0
wsdisplay0 at macfb0 (kbdmux ignored): console (std, vt100 emulation)
sn0 at obio0: integrated Ethernet adapter
sn0: Ethernet address 08:00:07:9c:a4:16
iwm0 at obio0: Apple GCR floppy disk controller
fd0 at iwm0 drive 0: (drive empty)
zsc0 at obio0 chip type 0
zsc0 channel 0: d_speed   9600 DCD clk 0 CTS clk 0
zstty0 at zsc0 channel 0
zsc0 channel 1: d_speed   9600 DCD clk 0 CTS clk 0
zstty1 at zsc0 channel 1
nubus0 at mainbus0
ae0 at nubus0 slot d: MacCon Ethernet, 64KB memory
ae0: Ethernet address 00:00:94:07:c4:c4
fpu0 at mainbus0 (mc68040)
scsibus0: waiting 2 seconds for devices to settle...
sd0 at scsibus0 target 0 lun 0: <IBM, DCAS-34330, S65A> SCSI2 0/direct fixed
sd0: 4134 MB, 8205 cyl, 6 head, 171 sec, 512 bytes/sect x 8467200 sectors
sd0: sync (160.0ns offset 15), 8-bit (6.250MB/s) transfers, tagged queueing
adb0 (direct, II series): 2 targets
aed0 at adb0 addr 0: ADB Event device
akbd0 at adb0 addr 2: extended keyboard (ISO layout)
wskbd0 at akbd0 (mux ignored): console keyboard, using wsdisplay0
ams0 at adb0 addr 3: 1-button, 100 dpi mouse
wsmouse0 at ams0 (mux ignored)
boot device: sd0
root on sd0a dumps on sd0b
root file system type: ffs
```

Sobald das System ordnungsgemäß gebootet ist, müssen Sie die Grundkonfigurationsdateien anpassen. Entscheidend sind die Dateien »/etc/fstab« und »/etc/rc.conf«. Laden Sie zuerst die »rc.conf« in den *vi* (ein anderer Editor steht momentan nicht zur Verfügung) und ändern beziehungsweise ergänzen Sie die folgenden Eintragungen:

```
rc_configured=YES
dhclient=YES
```

Die Angabe von *dhclient* macht natürlich nur Sinn, wenn im Rechner eine Netzwerkkarte eingebaut ist und außerdem ein DHCP-Server verfügbar ist. Falls kein DHCP-Server erreicht werden kann, muß die Netzwerkkarte manuell mit IP-Adresse und Netzmaske eingestellt werden (siehe Kapitel 7.1.1).

In der Datei »/etc/fstab« müssen jetzt die zusätzlichen und momentan nicht eingebunden FFS-Partitionen eingebunden werden.

```
/dev/sd0a       /               ffs     rw 1 1
/dev/sd0b       none            swap    sw 0 0
kern            /kern           kernfs  rw 0 0
proc            /proc           procfs  rw 0 0
/dev/sd0g       /usr/X11R6      ffs     rw 1 1
/dev/sd0f       /usr/pkgsrc     ffs     rw 1 1
/dev/sd0e       /usr/pkg        ffs     rw 1 1
```

Die Einträge für Root, Swap, Kernel-und Proc-Dateisystem sind vom Installationsprogramm bereits angelegt worden, die restlichen Einträge müssen von Hand erfolgen. Leider ist es nicht möglich, Macintosh-HFS-Partitionen einzubinden.

Sie werden jetzt bestimmt zum ersten Mal versuchen, die deutsche Tastatur einzuschalten. Das geht leider nicht! Obwohl NetBSD 1.6 auch auf dem Macintosh inzwischen die WS-Console unterstützt, ist keine Tastaturumschaltung möglich. Der Versuch, dies zu tun, mündet nur in einer Fehlermeldung:

```
wsconsctl -w encoding=de
wsconsctl: WSKBDIO_SETENCODING: Invalid argument
```

Auf einer deutschen Tastatur ist es natürlich nicht besonders angenehm, mit amerikanischer Tastaturbelegung arbeiten zu müssen. Die Tastencodes beziehungsweise die den deutschen Tasten unterlegten Zeichen bei amerikanischer Tastaturbelegung finden Sie in Tabelle 2.4.

Prinzipiell ist es auch möglich, ohne eingebaute deutsche Tastaturbelegung die wichtigsten Tasten umzumappen. Das ist jedoch eine enorme Fleißarbeit, die sich offensichtlich noch niemand angetan hat. Leider hat die Tastaturbelegung auch Auswirkungen auf die der grafischen Oberfläche. X besitzt im Gegensatz beispielsweise zum Intel-Port keinen eigenen Tastaturtreiber, so daß sich die Tastaturbelegung auch dort auswirkt.

Prinzipiell ist damit die Installation abgeschlossen. Sie können jetzt über das Paketsystem externe Software einspielen oder versuchsweise den X-Server mit *startx* aufrufen. Gelingt letzteres nicht und Sie erhalten nur eine nichtssagende Fehlermeldung

```
X connection to :0.0 broken (explizit kill or server shutdown)
```

liegt das daran, daß X- und Textmodus-Auflösung nicht miteinander harmonieren. Das Problem von XFree86 auf dem Mac68k ist, daß mit der eingebauten Grafikkarte und dem normalen X-Server für den Mac-68k nur eine Grafikauflösung von 640x480 Pixeln in Schwarzweiß (also 1 Bit Farbtiefe) möglich ist. Wenn Sie die NetBSD-Konsole mit einer anderen Auflösung verwenden, läßt sich der X-Server nicht starten und zeigt nur eine Fehlermeldung an.

Sie sollten zum Schluß auch noch den Hostnamen, den Domainnamen und das Root-Passwort definieren. Es schadet auch nicht, User anzulegen. Die Programme dafür sind identisch zu den anderen Ports. *useradd* dient dem Anlegen der User, mit *passwd* richtet

man das Passwort ein und für den Hostnamen legt man eine Datei »/etc/myname« an, die nur den Hostnamen (in Kurzform) enthält. Das Anlegen eines Users (im folgenden mit dem Namen »joerg«) besteht aus dem Aufruf des Programms *useradd* mit einigen recht logischen Parametern:

```
#useradd -b /home -G wheel -g wheel -s /usr/pkg/bin/bash -m \
-d /home/joerg -k /etc/skel -u 1000 -v joerg
Command: /bin/mkdir -p /home/joerg
Command: cd /etc/skel && /bin/pax -rw -pe -v . /home/joerg
/home/joerg/.
/home/joerg/./.cshrc
/home/joerg/./.mailrc
/home/joerg/./.profile
/home/joerg/./.rhosts
Command: /usr/sbin/chown -R -h 1000:0 /home/joerg
Command: /bin/chmod -R u+w /home/joerg
#
```

Außerdem sollte gleich anschließend das Passwort des Users definiert werden:

```
#passwd joerg
Changing local password for joerg
New password:
Please enter a longer password.
New password:
Retype new password:
#
```

und *chpass* für die Korrektur der Userdaten aufgerufen werden:

```
#chpass joerg
```

Die Userdaten werden in den *vi* (oder einen anderen Editor, der in der Umgebungsvariable *$EDITOR* eingetragen ist) geladen:

```
#Changing user database information for joerg
Login: joerg
Password: TF6SWaq33KcOU
Uid [#]: 1000
Gid [# or name]: 0
Change [month day year]:
Expire [month day year]:
Class:
Home diretory: /home/joerg
Shell: /usr/pkg/bin/bash
Full Name:
Location:
Office Phone:
Home Phone:
~
~
~
~
~
~
~
~
~
/etc/pw.28269a: unmodified: line 1
```

Gegebenenfalls sollte man noch überprüfen, ob der neue User wirklich in der Datei »/etc/group« in die Gruppe *wheel* eingetragen wurde. Das ist dann nicht der Fall, wenn der Parameter *-G wheel* vergessen wurde. Dann muß er mit einem Komma getrennt hier eingetragen werden.

Die Domain teilt man dem System über die Datei »/etc/defaultdomain« mit. Auch hier wird nur der Domainname selbst angegeben. Den kompletten Rechnernamen ermittelt das System aus den Angaben in diesen beiden Dateien. Die Userangaben paßt man mit dem Program *chpass* an. Hier trägt man eine andere Shell oder auch den richtigen Namen des Users ein.

Was Sie sonst an Programmen benötigen, können Sie sich entweder als Binärpaket vom NetBSD-FTP-Server laden oder – wenn Sie viel Geduld haben – aus den Package-Sources selbst kompilieren. Dafür müssen aber auf jeden Fall die Installations-Sets »comp.tgz«, »xcomp.tgz« und »text.tgz« eingespielt sein. Da es keine aktuellen Binärpakete für NetBSD/mac68k 1.6 gibt und man auf die alten Packages von NetBSD 1.5 zurückgreifen muß, ist das Selbstkompilieren aber durchaus nicht abwegig.

Ob man mit einem 25 MHz Apple Macintosh und NetBSD wirklich glücklich wird, soll an dieser Stelle nicht diskutiert werden. Eine Alternative zu einem alten 386-er oder einem lowend 486-er PC ist es schon. Mit moderner Hardware kann das System aber nicht konkurrieren. Einiges, wie die Einbindung ins Netzwerk funktioniert mit NetBSD sehr viel problemloser als mit dem original MacOS, anderes wie beispielsweise die Gra-

fik ist ausgerechnet auf einem Macintosh jämmerlich. Gerade die mangelhafte Grafik und die unzureichende Tastatursteuerung lassen diesen NetBSD-Port auf den ersten Blick eher fragwürdig erscheinen.

1	"	§	$	%	&	/	(	)	=	?	`
!	@	#	$	%	^	&	*	(	)	_	+
1	2	3	4	5	6	7	8	9	0	ß	´
1	2	3	4	5	6	7	8	9	0	-	=
Q	W	E	R	T	Z	U	I	O	P	Ü	*
Q	W	E	R	T	Y	U	I	O	P	{	}
q	w	e	r	t	z	u	i	o	p	ü	+
q	w	e	r	t	y	u	i	o	p	[	]
A	S	D	F	G	H	J	K	L	Ö	Ä	'
A	S	D	F	G	H	J	K	L	:	=	|
a	s	d	f	g	h	j	k	l	ö	ä	#
a	s	d	f	g	h	j	k	l	;	'	\
>	Y	X	C	V	B	N	M	;	:	_	
~	Z	X	C	V	B	N	M	<	>	?	
<	y	x	c	v	b	n	m	,	.	-	
`	z	x	c	v	b	n	m	,	.	/	

Tabelle 2.4: Amerikanische Belegung auf der deutschen Tastatur. Oben (grau hinterlegt) die jeweilige deutsche Taste, darunter das resultierende Zeichen

Es gibt aber – wenn auch externe – Abhilfe für einen Teil der Probleme. Was benötigt wird, sind ein anderer X-Server und eine deutsche Tastaturbelegungstabelle. Beides gibt es, die Tastaturbelegungsdatei, die allerdings nur für X gilt – auf der Konsole ist so ohne weiteres keine deutsche Tastaturbelegung möglich – befindet sich auf dem NetBSD-ftp-Server unter *ftp://ftp.netbsd.org/pub/NetBSD/arch/mac68k/contrib/Xmodmap/de*. Es sind hier zwei Dateien abgelegt, »Apple_Extended_Keyboard.Xmodmap« und »Apple_Extended_Keyboard_II.Xmodmap«. Erstere ist für Rechner mit normaler Tastatur, die zweite für solche, die eine Apple-Extended-Tastatur besitzen, bei der sich die Taste <> zwischen [Shift] und [Y] befindet. Zu diesen Rechnern gehört auch der als Installationsmaschine genutzte Quadra 700.

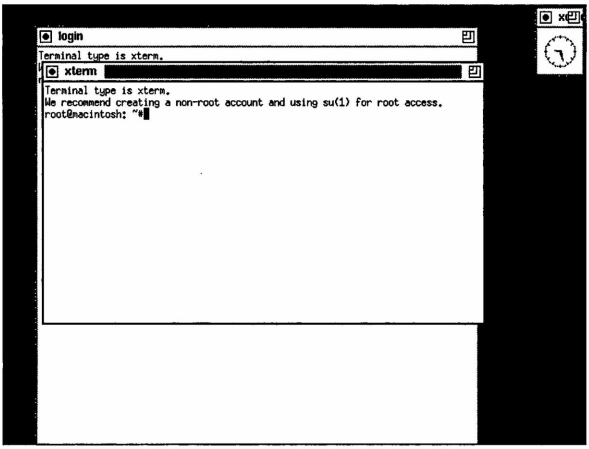


Abb. 2.30: twm mit dem Monochrom-X-Server

Der bessere X-Server befindet sich auf *ftp://ftp.netbsd.org/pub/NetBSD/arch/mac68k/contrib/Xmac68kColor-elf/*. Von den Dateien im Verzeichnis wird eigentlich nur »Xmac68k_color.bz2« benötigt. Dieses Archiv wird mit

```
#bunzip2 Xmac68k_color.bz2
```

ausgepackt und die Datei »Xmac68k_color« nach »/usr/X11R6/bin« kopiert. Die anderen Dateien sind Quelltexte und eine Kurzdokumentation. Die Dokumentationen sollte man lesen, die Quellen braucht man nicht.

Nach dem Download muß der Server Ausführungs-Attribute erhalten und nach »X« gelinkt werden:

```
#rm X
#chmod 777 Xmac68k_color
#ln -s Xmac68k_color X
```

Beim neuen X-Server ist zu beachten daß er keine Modus-Umschaltung beherrscht. Das heißt, es wird immer nur eine Auflösung mit gleichbleibender Farbtiefe verwendet. Es kommen dabei die Daten des MacOS zum Einsatz, von dem aus NetBSD gestartet wird.

Um die deutsche Tastaturbelegung zu verwenden, kopiert man die passende Tastaturbelegungsdatei in ein sinnvolles Verzeichnis, beispielsweise »/usr/X11R6/lib/X11/xinit«, verkürzt den Namen auf ».Xmodmap«und ruft dann das Programm *xmodmap* testweise auf, wobei die Konfigurationsdatei als Parameter übergeben wird:

```
#xmodmap /usr/X11R6/lib/X11/xinit/.Xmodmap
```

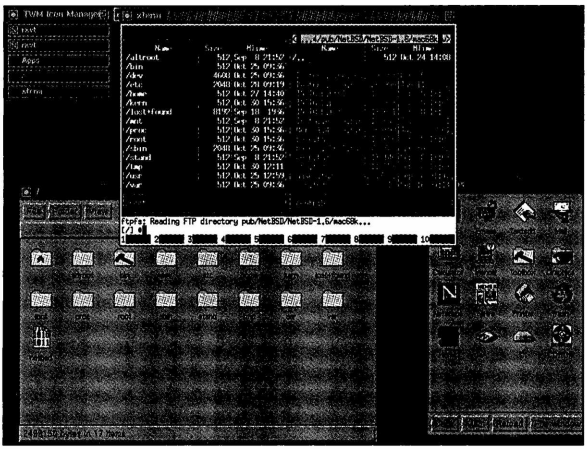


Abb. 2.31: Der twm mit dem alternativen X-Server

Maustastenproblem?

Die grafische Oberfläche auf Unix-Systemen erwartet normalerweise eine Drei-Tasten-Maus. Auf Systemen mit Zwei-Tasten-Mäusen simuliert man deshalb die dritte Taste durch einen gleichzeitigen Druck der beiden vorhandenen Buttons. Das geht natürlich mit der Ein-Tasten-Maus des Macintosh nicht. Diverse Programme – und insbesondere einige Window Manager – verlangen aber die rechte Maustaste. Ohne sie ist beispielsweise der WindowMaker nicht bedienbar, da damit das Menü aufgeklappt wird.

Eigentlich wäre es logisch gewesen, daß die beiden fehlenden Tasten durch gleichzeitiges Drücken des Mausbuttons und der Steuer-, Alt-, oder Apfel-Taste simuliert würden. Aber weit gefehlt! Wenig intuitiv wird die mittlere Taste im WindowMaker durch Drücken von [F11], die rechte Taste durch einen Druck auf [F12] simuliert, andere WindowManager unterstützen sie gar nicht.

xmodmap wertet die Datei aus und kehrt dann zurück. Sofort ist unter X die deutsche Tastatur eingestellt. Der Vorteil der hier gezeigten Benennung ist, daß bei Vorhandensein dieser Datei das Startskript die Datei automatisch lädt:

```
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap
```

```
# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# start some nice programs
twm &
xclock -geometry 50x50-1+1 &
xterm -geometry 80x20+494+51 &
xterm -geometry 80x20+494-0 &
exec xterm -geometry 80x20+0+0 -name login
```

Es handelt sich hier um die Original-Startdatei, in der nur die Anzahl der Zeilen der automatisch gestarteten X-Terminals auf sinnvollere Werte geändert wurden.

Wenn Sie einen anderen Window Manager lokal mit einer eigenen »~/.xinitrc« starten, müssen Sie hier natürlich die oben angegebenen Zeilen ergänzen.

Zum Schluß empfiehlt es sich, gleich den xdm einzuschalten, damit man mit der Konsole beim Start nicht mehr in Berührung kommt. Dazu ergänzt man die Datei »/etc/rc.conf« um die folgende Zeile:

```
xdm=YES
```

Um die deutsche Tastatur auch bei dieser Art des Starts der grafischen Oberfläche verwenden zu können, muß die Startdatei »/usr/X11R6/lib/ X11/xdm/Xsession« noch um den Tastaturbelegungsaufruf ergänzt werden. Die Datei »xinitrc« wird nämlich vom xdm nicht ausgewertet:

```
#!/bin/sh
# $XConsortium: Xsession /main/10 1995/12/18 18:21:28 gildea $

# redirect errors to a file in user's home directory if we can
```

```
for errfile in "$HOME/.xsession-errors" \
               "${TMPDIR-/tmp}/xses-$USER" \
               "/tmp/xses-$USER"
do
  if ( cp /dev/null "$errfile" 2> /dev/null )
  then
    chmod 600 "$errfile"
    exec > "$errfile" 2>&1
    break
  fi
done

xmodmap /usr/X11R6/lib/X11/xinit/.Xmodmap

case $# in
1)
  case $1 in
  failsafe)
    exec xterm -geometry 80x24-0-0
    ;;
  esac
esac

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ -f "$startup" ]; then
  exec "$startup"
else
  if [ -f "$resources" ]; then
    xrdb -load "$resources"
  fi

  case $# in
  1)
    case $1 in
    kde)
      exec $SHELL -c startkde
      ;;
    esac
  esac
  xclock -geometry 50x50-1+1 &
  xterm -geometry 80x25+0+0 &
  exec twm
fi
```

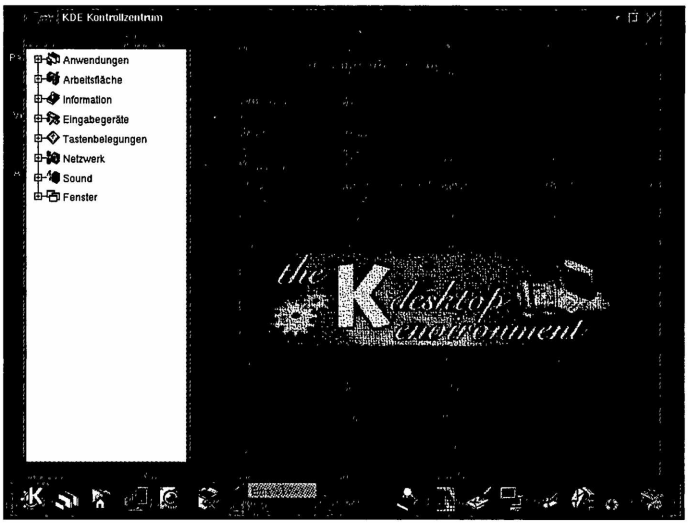


Abb. 2.32: KDE 1.1.2 unter NetBSD/mac68k und dem Farb-X-Server

Die Tastaturbelegung der Konsole soll auch angepaßt werden können. Wenn man in den NetBSD-Maillisten sucht, findet man die folgende Aussage:

```
Subject: Re: Swap and foreign keyboards ?
To: None <port-mac68k@netbsd.org>
From: Yasuhiro Endoh <quiver@tky3.3web.ne.jp>
List: port-mac68k
Date: 05/17/1999 07:25:50 Hello, I'm Endoh.

"Niels S. Eliasen" <nse@ruc.dk> wrote
> 2. How do I configure the system to have a non-US keyboard ?
>
> regards/mvh

I have Japanese keyboard(JIS keyboard). To use it on console, I
change kernel source. Keymap of console is written in
/usr/src/sys/arch/mac68k/include/keyboard.h

If you use non-US keyboard on console,
you must change keyboard.h and rebuild kernel.1
```

[1] Die Datei »keyboard.h« enthält tatsächlich die (amerikanische) Tastaturbelegungstabelle. Eine andere Belegung einzutragen ist mühsam, aber möglich.

```
On X, then you can change keymap by xmodmap.
See man xmodmap.
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
Yasuhiro Endoh quiver@tky3.3web.ne.jp
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
```

Das Neukompilieren des Kernels auf einer CPU mit nur 25 MHz ist aber ein sehr langwieriger Prozeß, deshalb ist es bequemer, auf die Konsole ganz zu verzichten und direkt in die grafische Oberfläche zu booten. Außerdem müssen die Quellen des Betriebssystems eingespielt werden, was reichlich freien Platz auf der Festplatte voraussetzt. Ob einem die deutsche Tastatur das wirklich wert ist, muß jeder für sich selbst entscheiden. Hinzu kommt, daß der Pseudo-Textmodus von NetBSD/mac68k technisch bedingt keine virtuellen Konsolen kennt, man also ohnehin auf externe Tools wie *screen* angewiesen ist, wenn man mehrere Konsolen zur Verfügung haben will.

2.3 Die Installation auf dem Amiga

von Ignatios Souvatzis, NetBSD/Amiga »port maintainer«

Dieser Abschnitt soll Sie mit den Besonderheiten der Systeminstallation auf einem Amiga bekannt machen. Er erläutert Systemvoraussetzungen, Besonderheiten der Installation und schließt mit amiga-spezifischen Tips zur Systemkonfiguration ab.

Die Installation von NetBSD auf Amiga unterscheidet sich von der *sysinst*-basierten menügeführten Installation auf i386-PCs und einigen anderen Architekturen dadurch, daß sie einen älteren skriptbasierten Installer benutzt. Deshalb soll sie relativ ausführlich beschrieben werden.

2.3.1 Systemvoraussetzungen

NetBSD-1.6 läuft auf allen Amiga-Rechnern mit einem Motorola M68020, M68030, M68040 oder M68060-Prozessor (CPU) und einer Speicherverwaltungseinheit (MMU), sowie auf dem DraCo[1].

Auf M68040- und M68060-CPUs ist auch eine Fließkommaeinheit (FPU) erforderlich – also die volle M68040- beziehungsweise M68060-CPU, da die Fließkommaemulation auf M68LC040, M68040V und M68LC060 fehlerhaft ist. Auf M68020- und M68030-CPUs wird die FPU jedoch weitestgehend emuliert, so daß wenigstens Systemprogramme korrekt ablaufen. M68020-Boards müssen die M68851-MMU enthalten. Mindestens die folgenden Amiga-Modelle sind ohne eine passende CPU-Karte also nicht NetBSD-fähig: A500, A600, A1000, A1200, A2000, A4000/EC030, CDTV und CD32.

Mindestspeichergröße sind acht Mbyte FastMem. Zorro-II-Speicherkarten im Amiga 3000 und Amiga 4000 funktionieren allerdings mit NetBSD nicht und dürfen ebenso-

[1] Es gibt gerüchteweise einige wenige DraCo-Exemplare mit einer M68040V-CPU; dort würde NetBSD auch nicht laufen.

wenig wie »Chip Memory« mitgerechnet werden, auch wenn sie formal »fast« sind. Der Speicher muß bei A3000/A4000 auf der Hauptplatine oder der CPU-Karte sitzen.

Vorsicht

Es gibt einige M68EC030, die *Teile* einer MMU implementiert haben. Das heißt, einige MMU-Register sind vorhanden. Manche »Testprogramme« glauben, eine MMU zu sehen. Aber lassen Sie sich nicht täuschen. Wenn bei installiertem Enforcer (aus dem Amiga Developer-Toolkit) dessen Testprogram »Lawbreaker« keine Traps erzeugt oder wenn NetBSD nicht booten will, ist es kein echter M68030-Prozessor. Leider wurden diese Prozessoren in einigen alten Beschleunigerkarten verbaut. Für AmigaOS sind sie in Ordnung, aber eine MMU-basiertes Betriebssystem wie NetBSD kann damit leider nichts anfangen.

Unterstützte Plattencontroller sind fast alle; wichtigste Ausnahmen sind das SCSI-Modul von Blizzard III und BlizzardPPC sowie die 4-aus-2-IDE-Adapter. Die volle Liste sollten Sie auf der NetBSD-Webseite nachlesen[1].

PowerPC-Karten für den Amiga – genauer: die PowerPC-CPU – werden bislang nicht unterstützt. Allerdings sind die bisher vorhandenen Karten (Cyberstorm PPC und BlizzardPPC) Doppelprozessorkarten, das heißt, sie enthalten auch eine CPU des Typs Motorola M68040 beziehungsweise M68060, auf der NetBSD selbstverständlich läuft. Die Benutzung der PowerPC-CPU wird NetBSD/amigappc übernehmen, das sich zur Zeit jedoch noch im Entwicklungsstadium befindet.

2.3.2 Vorbereitung der Installation

Die Vorbereitung der Installation umfaßt drei Schritte: erstens eine Datensicherung noch gebrauchter Daten der Platte, zweitens müssen Sie die Festplatte beziehungsweise eine Ihrer Festplatten unter AmigaOS partitionieren, drittens müssen Sie sich für eine Methode entscheiden, die Installationsarchive auf Ihren Rechner zu übertragen. Zur Vorbereitung sollte auch gehören, sich »INSTALL.ps«, »INSTALL.txt« oder »INSTALL.html« auszudrucken und den Ausdruck bei der Installation bereitzuhalten, außer Sie haben einen zweiten Rechner, den Sie zum Lesen benutzen können.

Datensicherung

Es sollte eine Selbstverständlichkeit sein: Man sollte immer von seinem System eine Sicherung besitzen und sie auch zurückspielen können. Ist das der Fall? Funktionieren Ihre AmigaOS-Installations-CD und Ihre Notfalldiskette (bei OS3.5 und 3.9) beziehungsweise Installationsfloppy (DraCo) noch? Oder haben sie ein funktionierendes Notsystem auf einer schreibgeschützen MO?

[1] Die Anlaufstelle ist *http://www.de.netbsd.org/de/Ports/amiga/*.

```
program can cause SIGNIFICANT data loss, and you are advised
to make sure your data is backed up before beginning the
installation process.

Default answers are displayed in brackets after the questions.
You can hit Control-C at any time to quit, but if you do so at a
prompt, you may have to hit return.  Also, quitting in the middle of
installation may leave your system in an inconsistent state.

Proceed with installation? [n] y
Cool!  Let's get to it...

The installation program needs to know which disk to consider
the root disk.  Note the unit number may be different than
the unit number you used in the standalone installation
program.

Available disks are:

sd0
sd1
sd2

Which disk is the root disk? sd2
```

Abb. 2.33: Die Auswahl der Festplatte

Normaler Betrieb kann ein System immer schädigen – stellen Sie sich einen plötzlichen Stromausfall während einer Schreiboperation vor. So etwas kann eine Festplatte irreparabel schädigen oder zumindest eine Neuformatierung erforderlich machen.

Die Installation eines Zweitsystems kann sehr wohl durch einen Irrtum oder Bedienungsfehler die falsche Platte überschreiben, was mindestens genauso katastrophal sein kann.

Als letzte Maßnahme nach dem letzten Backup sollten Sie sich die alte Partitionsdaten notieren. Aber auf Papier bitte!

Partitionierung

NetBSD/Amiga benutzt wie AmigaOS auch die RDB-Struktur. Da NetBSD aber über keinen Editor für diese Datenstruktur verfügt, müssen Sie die Partitionen für NetBSD unter AmigaOS anlegen, zum Beispiel mit *HDToolBox*. Bitte machen Sie sich mit der Bedienungsanleitung des Partitionstools vertraut und halten Sie sie zur Hand.

Empfohlen werden mindestens die folgenden Partitionen:

Stringform	Hexform	NetBSD	Kommentar
NBR\7	0x4e425207	/	Root-Partition
NBS\1	0x4e425301	(swap)	Swap-Partition
NBU\7	0x4e425507	z.B. /usr	Weitere Daten, BSD FFS

Weitere Partitionen machen beispielsweise für »/var« und »/usr/home« Sinn, wenn die Festplatte groß genug ist. Falls Sie experimentierfreudig sind, können Sie eine LFS-Partition anlegen, in obiger Tabelle also:

NBU\9	0x4e425509	z.B. /var/spool/news	BSD Logging File System

Wobei LFS aber immer noch als experimentell angesehen werden muß und eher für bestimmte Sorten flüchtiger Daten geeignet ist, wenn die Partition nicht sehr voll wird. Wenn Sie es nicht gezielt ausprobieren wollen: beschränken Sie sich auf NBU\7. Keine der Partitionen für die Systemverzeichnisse (»/«, »/bin«, »/dev«, »/etc«, »/var«, »/usr«) sollte von vornherein als LFS-Dateisystem angelegt sei, da das Installationsskript damit nicht zurecht kommt.

Geben Sie der Root-Partition einen treffenden Namen wie beispielsweise »NetBSD-1.6«; das erleichtert das Auswählen des Betriebssystems, wenn Sie mehrere installieren. Noch wichtiger: Geben Sie der Swap-Partition einen treffenden Namen. Sie werden ihn später brauchen, wenn Sie das Installationsdateisystem darauf kopieren und dann noch einmal, wenn Sie es starten.

```
Available are sd0 sd1 sd2. Look at which? [done]

As a reminder: the 'c' partition is assigned to the whole disk and can't
normally be used for a any file system!


You will now have the opportunity to enter filesystem information.
You will be prompted for device name and mount point (full path,
including the prepending '/' character).

Note that these do not have to be in any particular order.  You will
be given the opportunity to edit the resulting 'fstab' file before
any of the filesystems are mounted.  At that time you will be able
to resolve any filesystem order dependencies.

The following will be used for the root filesystem:
        sd2a    /

Device name? [RETURN if you already entered all devices] sd2d
Mount point? /usr

Device name? [RETURN if you already entered all devices]
Ok to configure sd2b as a swap device? [] yes

You have configured the following devices and mount points:

sd2a    /
sd2d    /usr
sd2b    swap

Filesystems will now be created on these devices.  If you made any
mistakes, you may edit this now.  Edit? [n] []
```

Abb. 2.34: Die Zuordnung der Partitionen zu Mountpoints

Im allgemeinen gilt:

- NBR\7 darf pro Platte nur einmal angelegt sein und wird von NetBSD auf die a-Partition abgebildet. Sie ist vom Typ »4.2BSD FFS« (beziehungsweise eine neuere Variante davon).
- NBS\1 darf pro Platte nur einmal vorhanden sein und wird von NetBSD auf die b-Partition abgebildet und ist vom Typ »Swap«.
- Die NetBSD-c-Partition einer Platte umfaßt die ganze Platte und ist von undefiniertem Typ.

```
Which disk is the root disk? sd2
If you like, you can now examine the labels of your disks.

Available are sd0 sd1 sd2. Look at which? [skip this step] sd2

# /dev/rsd2c:
type: SCSI
disk: DD640R2
label: part2
flags:
bytes/sector: 512
sectors/track: 25
tracks/cylinder: 33
sectors/cylinder: 825
cylinders: 541
total sectors: 446325
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

4 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a:     41250       825     4.2BSD   1024  8192     0   # (Cyl.    1 - 50)
 b:     41250     42075       swap                      # (Cyl.   51 - 100)
 c:    446325         0     unused      0     0         # (Cyl.    0 - 540)
 d:    363000     83325     4.2BSD   1024  8192     0   # (Cyl.  101 - 540)
disklabel: boot block size 0
disklabel: super block size 0

Available are sd0 sd1 sd2. Look at which? [done]
```

Abb. 2.35: Auswahl der Festplatte und Kontrolle der Partitionierung

NBU\7, NBU\9 und alle weiteren werden in der Reihenfolge ihres Auftretens auf die Partitionen »d« bis »p« abgebildet. Dabei ist NBU\7 vom Typ UFS, NBU\9 vom Typ LFS. Im allgemeinen ist bei NBR, NBS und NBU das letzte Byte der numerische NetBSD-Typ aus der Datei »/usr/include/sys/disklabel.h« (die FS_...-Konstanten).

Alle NetBSD-Partitionen werden mit 0 (Null) reservierten Blöcken am Anfang und am Ende angelegt! Der Bequemlichkeit und natürlich auch Kompatibilität mit verschiede-

nen AmigaOS-Dateisystemen und Linux wegen werden jedoch beispielsweise DOS\0 bis DOS\3 als AmigaOS-OFS beziehungsweise FFS erkannt, LNX\0 als ext2fs von Linux und MSD\0 als MS-DOS-Dateisystem. Auf diese Weise können solche Partitionen von NetBSD mitbenutzt werden (AmigaOS-Filesysteme können zur Zeit aber nur lesend eingebunden werden).

```
You have configured the following devices and mount points:

sd2a    /
sd2d    /usr
sd2b    swap

Filesystems will now be created on these devices.  If you made any
mistakes, you may edit this now.  Edit? [n]
Creating filesystems...
/dev/rsd2a:     41250 sectors in 50 cylinders of 33 tracks, 25 sectors
        20.1MB in 1 cyl groups (112 c/g, 45.12MB/g, 5120 i/g)
super-block backups (for fsck -b #) at:
 32,
newfs: ioctl (WDINFO): Invalid argument
newfs: /dev/rsd2a: can't rewrite disk label

/dev/rsd2d:     363000 sectors in 440 cylinders of 33 tracks, 25 sectors
        177.2MB in 5 cyl groups (96 c/g, 38.67MB/g, 8832 i/g)
super-block backups (for fsck -b #) at:
     32,  79264, 158496, 237728, 316960,
newfs: ioctl (WDINFO): Invalid argument
newfs: /dev/rsd2d: can't rewrite disk label
```

Abb. 2.36: Dateisysteme werden initialisiert

Platten größer 4 Gbyte

Da die Swap-Partition beim Installieren und die Rootpartition im Betrieb zum Booten benutzt werden und die entsprechenden ROM-Routinen mit 32 Bit adressieren, müssen diese Partitionen *unbedingt* innerhalb der ersten vier Gbyte einer Festplatte liegen! Für die anderen Partitionen gilt diese Einschränkung nicht.

Bootblöcke

NetBSD benutzt einen Bootblock mit der Größe 8 Kbyte zum Booten. Bei herkömmlichen Platten (Blockgröße von 512 Byte) stellen Sie folgendes ein: *Custom Boot Code, 16 Blocks.* Bei *HDToolBox* finden Sie die nötigen Einsteller, wenn Sie Parameter einer Partition angeben, wenn sie *Advanced Options* aktivieren.

Die Bootpriorität der Swap-Partition sollte -1 sein, damit sie nicht über der von Nutzsystemen liegt und normalerweise nicht automatisch von dort gebootet wird. Zum Installieren wählen Sie später die Swap-Partition über das Amiga-Bootmenü aus. Die Bootpriorität der Rootpartition kann höher sein. Wenn sie größer ist als die irgendeiner anderen Partition, wird NetBSD – wenn Sie nicht über das Bootmenü eingreifen – automatisch davon booten. Sie sollte normalerweise aber kleiner als 5 sein, denn 5 ist die

hartkodierte Bootpriorität des Floppytreibers und Sie können dann nicht mehr ohne Eingriff über das Bootmenü von einer Floppy booten. Allerdings mag das auch ein Sicherheitsfeature sein, für den Fall, daß Sie virenverseuchte Floppy-Medien einzulegen pflegen.

Platten mit Blockgrößen ungleich 512 Bytes gibt es eigentlich fast nur in Form von MOs mit 640 Mbyte Kapazität und mehr. BSD-FFS und -LFS funktionieren dort zur Zeit noch nicht, aber AmigaOS-Filesysteme (lesend) oder beispielsweise Bandarchive (*tar*, *dump*) auf der rohen Platte können von NetBSD problemlos genutzt werden.

Randbemerkung

Partitionstools, die von der Firma Phase 5 bei ihren Blizzard- und Cyberstorm-Produkten beigelegt wurden, erlauben nicht die Konfiguration eines Bootblocks auf einer Partition und löschen das entsprechende Flag sogar, wenn eine vorhandene Partitionstabelle editiert wurde. Hier hilft folgender Trick:

Nachdem Sie die Partitionen selbst mit dem Phase5-Tool angelegt haben, benutzen Sie HDToolBox, um »Custom Boot Code«, die Anzahl der Bootcode-Blöcke und die Bootpriorität nachträglich einzustellen. Wenn Sie mit einem originalen AmigaOS-3.1-HDToolBox (oder älter) eine Platte größer 4 Gbyte bearbeiten, sollten Sie keine anderen Parameter verändern.

Bereitstellung der Installationsmedien

NetBSD/Amiga kann von verschiedensten Medien installiert werden, die die offizielle Installationsanleitung ausführt. Dabei dürften Floppy-Disks heutzutage eher unpraktisch sein.

Am handhabbarsten sind:

- CD-ROM: Eine offizielle oder inoffizielle CD-ROM, die den Teilbaum »pub/NetBSD/NetBSD-1.6« des offiziellen FTP-Servers oder eines Mirrors enthält oder aber zumindest die Unterverzeichnisse »amiga«, »shared/common« und »shared/m68k«, hat zwei Vorteile: Die Installation ist (hoffentlich) schneller als über die meisten Netzverbindungen und kann bei einem Schreibfehler während der Installation nicht zerstört werden.

- Ein AmigaOS- oder NetBSD-Filesystem: Wer den Amiga (auch) unter AmigaOS benutzen will und unter AmigaOS schon eine Netzverbindung konfiguriert hat, kann sich die Installationssets auf eine Amiga-FFS (nicht SFS, PFS und so weiter) kopieren und bei der Installation von dort aus benutzen.

  Wer schon eine NetBSD-Installation hat und auf eine weitere Platte installieren will oder die vorhandene Installation upgraden will, kann dies mit einer NetBSD-Partition ebenfalls tun.

- Ein FTP-Server: Wer eine stabile, hinreichend schnelle Netzverbindung besitzt, die ohne große Konfiguration funktioniert, kann direkt über eine ftp-Verbindung installieren. Das schließt Modem- und ISDN-Verbindungen aus, auch PPPoE dürfte ohne einen zwischengeschalteten Router ausgeschlossen sein. Damit bleiben direkte Ethernet- und ARCnet-Verbindungen übrig. Sie können dazu einen offiziellen FTP-Mirror von NetBSD benutzen oder die Daten vorher auf eine (zweite) eigene Maschine kopieren, wo Sie dann – zumindest während der Installation – einen FTP-Server laufen lassen, der das lokale Netzwerk bedient.

```
On some console hardware, such as serial consoles and Sun frame buffers,
this can extend the total extraction time.
Use verbose listing for extractions? [y]
Install from (f)tp, (t)ape, (C)D-ROM, (N)FS or local (d)isk? f

This is an automated ftp-based installation process.  You will be asked
several questions.  The correct set of commands will be placed in a script
that will be fed to ftp(1).

Server IP? [] 204.152.184.75
Login? [] ftp
Password?

You will be asked to enter the name of the directory that contains the
installation sets. When you enter a '?' you will see a listing of the
current directory on the server.
Server directory? [] /pub/NetBSD/NetBSD-1.6/amiga/binary/sets
The following sets are available for extraction:
(marked sets are already on the extraction list)

    kern-GENERIC.tgz
    base.tgz
    comp.tgz
    etc.tgz
    games.tgz
    man.tgz
    misc.tgz
    text.tgz
    xserver.tgz
    xbase.tgz
    xcomp.tgz
    xcontrib.tgz
    xfont.tgz
    xmisc.tgz

Continue to add filenames [y]?
```

Abb. 2.37: Die Installation über FTP wird ausgewählt

2.3.3 Die eigentliche Installation

Die eigentliche Installation besteht wieder aus mehrereren Schritten, im wesentlichen aus:

- Dem Kopieren des Installations-Filesystems auf die Swap-Partition,
- dem Booten des Installationssystems,
- der Auswahl der Zielplatte/-partitionen und der Installationsmedien und
- dem Übertragen der Daten.

Nun erfolgt das Kopieren des Installations-Filesystems auf die Swap-Partition. Unter einem laufenden NetBSD-, Linux- oder Amiga-Unix geht das mit dem Befehl *dd*

```
dd if=miniroot.fs of=/dev/sd0b bs=64k
```

beziehungsweise

```
gzip -dc miniroot.fs.gz | dd of=/dev/sd0b  bs=64k
```

Die benutzte Swap-Partition darf dabei nicht aktiv sein!

Unter AmigaOS setzen Sie ein Tool ein, das rohe Partitionen ansprechen kann. Eine NetBSD-Distribution liefert dazu im Verzeichnis »pub/NetBSD/NetBSD-1.6/amiga/installation/misc« das Programm *xstreamtodev* mit, das hier kurz beschrieben werden soll. Alternativ können Sie beispielsweise das frei erhältliche Tool *dcp* (Aminet) benutzen.

```
xstreamtodev --input=miniroot.fs --rdb-name=<swap partition>
```

Hier muß für *<Swap partition>* der AmigaOS-Name der Partition angegeben werden, den Sie oben vergeben haben!

Falls aus irgendeinem Grund *xstreamtodev* nicht in der Lage ist, die Partition zu finden, können Sie dem Programm mit den weiteren Parametern

```
--device=<driver.name>
```

zum Beispiel *--device=scsi.device*

```
--unit=<SCSI unit number>
```

zum Beispiel *--unit=0* eine Suchhilfe geben.

Das Booten von der Swap-Partition

Diese Beschreibung bezieht sich auf das AmigaOS 3.0 beziehungsweise 3.1 ROM. Bei älteren ROMs gibt es Abweichungen.

Halten Sie beide äußeren Maustasten beim Booten gedrückt, bis der Schirm *Amiga Early Startup Control* erscheint. (Beim DraCo müssen Sie stattdessen *erst nach Aufforderung* die linke Maustaste drücken.)

Falls das Bootbild nicht synchronisiert wird: Durch Drücken einer Taste auf der Tastatur können Sie beim Amiga den Videomodus verändern (50/60 Hz). Beim DraCo können Sie mit der Kombination [Alt-Links]+[Alt-Rechts]+[Help] die vorhandenen Videomodi durchprobieren.

Klicken Sie auf *Boot options*. Im neuen Schirm sehen Sie zwei Tabellen. Die erste heißt *Control Active Devices* und kann Sie unterstützen, die richtige Partition zu wählen, da hier neben dem Namen auch Partitionstyp, Treiber und Unitnummer angegeben sind.

Wenn Sie die richtige Partition ausgewählt haben – für die Installation also die vorhin beschriebene Swappartition – klicken Sie sie einmal in der *Select Boot Device*-Tabelle an. Hier können Sie für den Bootprozeß auch noch temporär den CPU-Cache ausschalten. Meist sollte das aber nicht nötig sein. Dann kehren Sie mit *Use* in den Startschirm zurück.

Nun können Sie durch Klicken auf *Boot* den Bootblock starten.

Sie erhalten eine Begrüßungszeile vom primären Bootblock und ohne sichtbare Zeitverzögerung eine Begrüßungszeile vom sekundären Bootblock. Darunter haben Sie wenige Sekunden Zeit, eine neue Kommandozeile anzugeben, die alte zu bestätigen, oder Sie warten einfach ab, bis es automatisch weitergeht.

Die normale Startzeile zum Installieren ist

```
netbsd -bsSn2
```

Es ist bei manchen Maschinen nötig, ohne Symbole (also ohne »S« im Optionsstring) und mit nur einem Speicherblock (ohne »n2«) zu booten.

Nun zeigt ihnen ein rotierender Strich den Vorgang des Ladens an, ergänzt durch die Größe der geladenen Kernel-Segmente. Dann wechselt der Videomodus, während der Kernel sich initialisiert, und Sie sehen die Bootmeldungen des Kernels. Da Sie mit »-b« gebootet haben, werden Sie gleich nach Root Device (*sd0b*), Root Filesystem (*ffs* beziehungsweise *generic*) und Dump Device gefragt (bei letzterem geben Sie nichts an).

Der Kernel läßt Sie nun eine Shell auswählen, drücken Sie nur die [Return]-Taste.

Auswahl der Tastaturbelegung

Als nächstes startet der Kernel das Installationsskript. Sie werden nach der Tastaturbelegung gefragt und können mit einer Ziffer einen Typ auswählen und mit [Return] bestätigen.

Nun können Sie zwischen *I)nstallation* und *U)pgrade* auswählen. Ich beschreibe im folgenden die Installation; ein Upgrade auf 1.6 ist wegen des Übergangs vom a.out- auf das ELF-Format bei den Programmen eine etwas kitzlige Angelegenheit und für Anfänger nicht unbedingt zu empfehlen. Sicherer ist es, auf eine zweite Platte zu installieren und persönliche Daten später herüberzukopieren. (Falls Sie es doch tun wollen: der größte Teil der Prozedur ist identisch. Lesen Sie daher erst den Rest dieser Anleitung, gegen Schluß gehe ich kurz auf die Probleme dabei ein.)

Sie geben also [I] ein, immer gefolgt von der [Return]-Taste.

Das Skript begrüßt Sie nun und warnt Sie, wie ich ein paar Abschnitte weiter oben, vor möglichen Konsequenzen einer fehlgeschlagenen Installation. Sie können ein letzes Mal mit [n] die Installation abbrechen. Danach gibt es kein Zurück mehr.

Auswahl der Platten und Partitionen

Sie bekommen nun die Liste der Platten angezeigt und können sich die Platte aussuchen, auf der die Rootpartition angelegt werden soll. Dazu sollten Sie folgendes wissen:

Der *GENERIC*- und der Installationskernel numerieren beide die Platten nach Bustyp durch, das heißt, die SCSI- und ATAPI-Platten heißen der Reihe nach *sd0*, *sd1*, *sd2* ... und die IDE-Platten heißen *wd0*, *wd1*. Die Numerierung entspricht nicht der absoluten SCSI-ID! Um die richtige Zuordnung zu treffen, sollten Sie die Kernelmeldungen mitlesen.

Bevor die Partitionen für die Installation zugeordnet werden, haben Sie noch die Gelegenheit, sich die NetBSD-Sicht der Partitionstabelle einer oder mehrerer Platten anzusehen; geben Sie hierzu einen der NetBSD-Plattennamen an, die Ihnen das Skript vorschlägt. Eine leere Eingabe führt an dieser Stelle weiter. Im folgenden nehme ich an, daß die Platte *sd0* benutzt werden soll.

Das Skript baut sich nun eine Liste der Dateisysteme für die Installation auf. Ohne Ihr Zutun ist schon zum Beispiel »sd0a /« in der Liste. Sie können nun weitere Partitionen, zum Beispiel »sd0d« und den Montagepunkt, beispielsweise »/usr«, angeben. Zum Schluß fragt Sie das Skript, ob »sd0b« als Swap-Device benutzt werden soll – hier können Sie bestätigen. Benutzen Sie die Partition C für nichts, das ist eine Pseudopartition zum Zugriff auf die ganze Platte.

Auf den ausgewählten Partitionen werden nun Dateisysteme angelegt.

Netzwerkkonfiguration

Sie werden nun nach Netzwerkparametern gefragt, in einer Schleife nach Netzwerkinterface getrennt, falls Sie mehrere installiert haben. Wenn Sie nicht über das Netz installieren wollen, können Sie mit einer leeren Eingabe zum nächsten Schritt vorspringen, sonst müssen Sie Maschinennamen, Domainnamen, IP-Adresse, Netzmaske, Adresse eines Nameservers und eines Routers angeben.

Sie haben nun eine letzte Gelegenheit, die Filesystemtabelle zu editieren, falls sie beispielsweise das Network File System oder schon vorhandene Filesysteme für die Installation montieren wollen.

Beginn der eigentlichen Installation

Sie können sich nun ein Installationsdevice aussuchen und E-Mail-Adresse oder irgendwas gefolgt von einem @ eingeben. Gegebenenfalls werden Sie nach weiteren Parametern gefragt, beispielsweise bei Installation von ftp nach dem Servernamen, user (oft *ftp*) und Passwort (in diesem Fall Ihre E-Mail-Adresse oder irgendwas gefolgt von einem @).

Außerdem können Sie sich Installationssets aussuchen. Installieren Sie mindestens *base* und *etc*. *man* enthält das Manual, *comp* den Compiler und Verwandte, *games* eine kleine Zahl meist traditioneller Spiele aus der Unix-Geschichte. *kern* ist ein GENERIC-Kernel, und die Sets mit X enthalten die Grundausstattung des X-Window-Systems.

```
[X] kern-GENERIC.tgz
[X] base.tgz
    comp.tgz
    etc.tgz
    games.tgz
    man.tgz
    misc.tgz
    text.tgz
    xserver.tgz
    xbase.tgz
    xcomp.tgz
    xcontrib.tgz
    xfont.tgz
    xmisc.tgz

Continue to add filenames [y]?
File name [comp.tgz]? etc.tgz
The following sets are available for extraction:
(marked sets are already on the extraction list)

[X] kern-GENERIC.tgz
[X] base.tgz
    comp.tgz
[X] etc.tgz
    games.tgz
    man.tgz
    misc.tgz
    text.tgz
    xserver.tgz
    xbase.tgz
    xcomp.tgz
    xcontrib.tgz
    xfont.tgz
    xmisc.tgz

Continue to add filenames [y]? n
```

Abb. 2.38: Auswahl der zu installierenden Betriebssystemkomponenten

Nachdem Sie die Sets ausgewählt haben, können Sie – besonders bei einer Netzwerkinstallation – das System erst einmal sich selbst überlassen.

Auswahl der Zeitzone

Das System fragt Sie nun nach der Zeitzone, die zur Zeitanzeige benutzt wird. »Europe/Berlin« sollte Ihren Ansprüchen genügen. Das System erzeugt nun die Geräteknoten, was wieder eine Weile dauert, und kopiert den Tastaturtyp. Sie haben hier die Gelegenheit, den Installationskernel statt des GENERIC aus dem *kern*-Installationsset auf ihr System zu kopieren. Er ist handlicher und bootet oft zuverlässiger.

```
./var/games/saillog
./var/games/snakerawscores
./var/games/snake.log
./var/games/tetris.scores
./var/log/aculog
./var/log/authlog
./var/log/lastlog
./var/log/lpd-errs
./var/log/maillog
./var/log/messages
./var/log/secure
./var/log/sendmail.st
./var/log/wtmp
./var/log/xferlog
./var/msgs/bounds
./var/run/utmp
./var/yp/Makefile.main
./var/yp/Makefile.yp
./.cshrc
./.profile
pax: ustar vol 1, 223 files, 692224 bytes read, 0 bytes written in 8 secs (86528
 bytes/sec)
226 Transfer complete.
119373 bytes received in 00:08 (14.27 KB/s)
221-
    Data traffic for this session was 19167931 bytes in 3 files.
    Total traffic for this session was 19171286 bytes in 3 transfers.
221 Thank you for using the FTP service on ftp.netbsd.org.
Extraction complete.

Extract more sets? [n] []
```

Abb. 2.39: Die Betriebssystemkomponenten sind ausgepackt

Nun sind sie (fast) fertig, und können in das neuinstallierte System booten. Auf die Eingabeaufforderung »#« geben Sie *reboot* dazu ein.

2.3.4 Nachbereitung

Erstmaliges Booten des installierten Systems

Das Booten erfolgt fast genauso wie das Booten der Swap-Partition.

Besitzt die NetBSD-Rootpartition die höchste Bootpriorität, brauchen Sie NetBSD nicht vom Bootmenü aus auszuwählen. Wenn der NetBSD-Bootblock nach der Kommandozeile fragt, können Sie

```
netbsd -s
```

oder

```
netbsd -sn2
```

angeben, um in den Single-User-Mode zu gelangen. Die Angabe von *-b* ist normalerweise nicht nötig, da NetBSD im Regelfall die Partition, von der der Bootblock geladen wurde, finden kann.

Filesystemcheck, Montage der Filesysteme

NetBSD läßt Sie wieder die Shell [/bin/sh] und den Terminaltyp [vt220] bestätigen, danach sehen Sie als Eingabeaufforderung ein »#«. Starten Sie den Filesystemtest durch

```
# fsck -p
```

(das # brauchen Sie nicht einzutippen) und mounten Sie anschließend die Filesysteme mit

```
# mount -av
```

Setzen das Root-Passworts

Anschließend sollten Sie sofort ein Passwort setzen, das dann für den User *root* gilt:

```
# passwd
```

Das Kommando fragt Sie sicherheitshalber zweimal nach dem neuen Passwort, denn Sie tippen es unsichtbar ein.

Sie können anschließend durch die Eingabe von *exit* das System weiterbooten lassen oder durch *reboot* neu booten. Wenn Sie jetzt beim Bootlader das »s« bei den Optionen weglassen, wird NetBSD bis in den Multiusermode booten. Sie können sich dann als *root* unter Angabe des eben gesetzten Passworts einloggen, um normale User einzurichten. Als *root* sollten Sie ausschließlich das System administrieren!

Weitere Kapitel dieses Buchs beschreiben die Administration von NetBSD genauer. Im folgenden werden nur die amiga-spezifischen Änderungen behandelt.

Aufsetzen des X Servers bei NetBSD/Amiga

Dies ist in englischer Sprache sehr gründlich im Dokument »INSTALL.X11« beschrieben. Im folgenden das wichtigste daraus.

Grafikdevice und weitere Parameter des X Servers

Der X Server *Xamiga* braucht zunächst die Angabe eines Grafikdevice. Es entspricht der benutzten Grafikkarte nach folgender Tabelle:

grf0	Eingebaute Amiga-Grafik	
grf1	Retina Z2	(wird vom X Server nicht bedient)
grf2	Retina Z3, DraCo Altais	
grf3	Picasso und ähnliche	
grf4	A2410	(wird vom X Server nicht bedient)
grf5	Cybervision64	
grf6	oMniBus, Domino, Merlin	
grf7	CyberVision64/3D	

Bei einigen Grafikkarten sind weitere Parameter nötig.

Bei *grf0* wäre es beispielsweise die folgende Zeile:

```
-dev grf0 -width 640 -height 480 -depth 4
```

-width ist die zu nutzende Bildschirmbreite in Pixeln, *-height* die Höhe, *depth* die Farbtiefe. Je nach Amiga-Modell und diesen Angaben wird ein passender Betriebsmodus ausgewählt. Für AGA-Modi bei A1200 und A4000 müssen Sie beim Booten des NetBSD-Kernels *-A* zusätzlich angegeben haben.

```
grf2,3,5-9      -mode N
```

Der Videomodus, wird bei der Retina aus einer festen Tabelle entnommen, bei den anderen Karten muß er vorher geladen werden (siehe unten).

-mouse /dev/mouse0	Normaler Mausport
-mouse /dev/mouse1	Maus im Joystickport
-mouse /dev/ttyC1	Serielle Maus (besonders am DraCo)
-mouse /dev/tty00	Serielle Maus im seriellen Port eines Amiga.

Die X-Kommandozeile sieht dann beispielsweise so aus:

```
in /usr/X11R6/lib/X11/xdm/Xservers
:0 local /usr/X11R6/bin/X -dev /dev/grf2 -mode 7
```

In »/usr/X11R6/lib/X11/xinit/xserverrc« oder in »~/.xserverrc« im Homeverzeichnis eines normalen Users:

```
Xwrapper -dev /dev/grf0 -width 640 -height 540
```

Tastaturbelegung für X11

Eine deutsche Tastaturbelegung wählen Sie durch diese Parameter aus:

```
+kb -xkbmap de
```

Falls sich keine Keymap in »/usr/X11R6/lib/X11/xkb/compiled/de.xkm« befindet oder Sie sie abändern wollen, führen Sie (als *root*) folgendes aus:

```
cd /usr/X11R6/lib/X11/xkb
xkbcomp -xkm -o compiled/de.xkm -m de keymap/amiga
mv /tmp/de.xkm /usr/X11R6/lib/X11/xkb/compiled/de.xkm
```

Diese Dateien enthalten die nötigen Definitionen, die Sie editieren müßten:

```
/usr/X11R6/lib/X11/xkb/geometry/amiga
/usr/X11R6/lib/X11/xkb/keycodes/amiga
/usr/X11R6/lib/X11/xkb/keymap/amiga
/usr/X11R6/lib/X11/xkb/symbols/amiga
```

Grafikmodus für einige Grafikkarten

Die Gerätetreiber grf3, 4, 5, 7 benutzen für den Grafikmodus einen ladbaren Videomodus. Seine Einstellung wird mit dem Programm *grfconfig* vorgenommen.

Erzeugen Sie eine Datei »/etc/rc.d/grfconfig« mit dem folgenden Inhalt:

```
#!/bin/sh

# PROVIDE: grfconfig
# REQUIRE: mountall
# BEFORE: xdm

required_files=/etc/grfmodes
stop_cmd=":"
start_cmd="loadmode"
loadmode() {
  echo 'loading grfmodes'
```

```
  /usr/sbin/grfconfig ${grfconfig_flags:-/dev/grf3} /etc/grfmodes
}
```

In »/etc/rc.conf« tragen Sie die folgende Zeile ein, wobei Sie gegebenenfalls »/dev/grf3« durch das richtige Device ersetzen:

```
grfconfig=YES grfconfig_flags=/dev/grf3
```

»/etc/grfmodes« enthält eine oder mehrere Zeilen, die folgende Spalten enthalten – lesen Sie mit *man grfconfig* die genaue Bedeutung nach:

```
num clk      wid hi  dep hbs hss hse ht  vbs vss vse vt  flags
```

Beispielsweise würden die folgenden drei Zeilen drei VGA-ähnliche Modi erzeugen, wovon der zweite das Zeilensprungverfahren nutzt und der dritte jede Zeile verdoppelt:

```
1   31000000 640 480  8  640 688 768 832 480 489 492 520 default
2   31000000 640 480  8  640 688 768 832 480 489 492 520 interlace
3   31000000 640 480  8  640 688 768 832 480 489 492 520 doublescan
```

Audio-Devices

Amigas können durchaus mehrere Audio-fähige Geräte enthalten, beispielsweise eingebauten Amigasound, (die originale) Melody Z2, Repulse und Toccata.

Diese werden in der Reihenfolge, wie sie erkannt werden, als */dev/audio0*, */dev/audio1* und so weiter ansprechbar; entsprechend die zugehörigen Sound-, Audioctl- und Mixer-Geräte.

Sie sollten Ihr bevorzugtes Gerät durch symbolische Links als Gerät ohne Nummer erreichbar machen, da manche soundnutzenden Programme sonst unglücklich sind, beispielsweise

```
cd /dev
ln audio1 audio
ln audioctl1 audioctl
ln sound1 sound
ln mixer1 mixer
```

Einen Haken gibt es zur Zeit bei der Repulse; ohne folgende Initialisierung hören Sie nichts, da die Initialisierung alle Kanäle stummschaltet:

```
mixerctl -w outputs.master.mute=off inputs.dac.mute=off
```

Vielleicht schreiben Sie dieses Kommando auch nach »/etc/rc.local«.

2.3.5 Upgrade statt Installation

Wenn Sie mutig sind, können Sie ein NetBSD-1.5-System auf NetBSD-1.6 upgraden. Sie geben dazu beim Booten des Installationssystems nach der Tastaturauswahl statt *I* ein *U* (für Upgrade) an.

Die nötigen Schritte sind ähnlich wie bei der Installation, allerdings entfällt das Erstellen der Filesysteme, da die alten Partitionen übernommen werden.

Eigentlich sollte nichts schlimmes passieren, selbst gemischte Systeme sind bedingt lauffähig, aber mit folgenden Einschränkungen:

- Der alte (einstufige) Bootblock und *loadbsd* können keine neuen ELF-Kernels laden.
- Ein alter Kernel kann neue Programme nicht ausführen.
- Ein neuer Kernel kannn alte Programme nur ausführen, wenn *options COMPAT_AOUT_M68K* in der Kernelconfiguration stand, der *GENERIC*- und der *INSTALL*-Kernel also nicht.
- Manche alten Dienstprogramme, etwa *netstat*, funktionieren mit neuen Kerneln gar nicht, da die Padding-Regeln für Strukturen zwischen a.out-Compiler und ELF-Compiler abweichen.
- Übriggebliebene a.out-Programme laufen oder nicht, was nicht wirklich problematisch ist.

  Übriggebliebene a.out-Libraries machen das Übersetzen neuer Programme zu einem Hindernislauf mit verbundenen Augen, da gemischtes Linken nicht funktioniert. Sie werden also nicht umhin kommen, Ihre gesamte Drittsoftware (auch im pkg-System) irgendwann neu zu übersetzen oder zu installieren.
- Wenn die alten Partitionen für NetBSD-1.6 zu knapp bemessen sind, bricht die Installation zwischendrin erfolglos ab; hoffentlich ist dann schon alles Nötige installiert, so daß das System lauffähig ist!

Wenn man davon absieht, können Sie also durchaus ein *U)pgrade* probieren. Aber beherzigen Sie meine Warnungen über vorhandene Backups bitte doppelt und dreifach!

2.4 Die Installation auf der Sparc

von Felix J. Ogris

Sparc-Prozessoren gibt es seit den achtziger Jahren. Die Firma Sun Microsystems entwickelte die sogenannte Scaleable Processor Architectur damals auf Basis der RISC-Architektur und fertigte mit Fujitsu die ersten sparc-basierten Systeme. Obwohl Sun 1989 die Spezifikationen für seine Prozessoren offenlegte und in die Hände der Sparc International getauften Organisation gab, gibt es neben dem Marktführer Sun nur wenige Hersteller von Sparc-Prozessoren: Neben den von Sun selbst produzieren SuperSPARC- und microSPARC-CPUs gibt es noch die HyperSPARC-Prozessoren von Ross Technology sowie die SPARClite-CPUs von Fujitsu. Ebenso sieht es bei den Komplettsystemen aus: Firmen wie Aries Research, Tatung und Tadpole bieten vor allem

sogenannte Sun-Clones an, also Rechner, die bis auf das Gehäusedesign den Workstations und Servern von Sun gleichen.

Wenn man heute also von einem Sparc- oder Solaris-System redet, ist in der Regel eine Sun-Maschine gemeint. 1993 brachte die Sparc International die Spezifikation für 64-bittige Prozessoren heraus (aka Sparc Version 9), die unter Marktnamen wie UltraSPARC von Sun oder SPARC64 von Fujitsu bekannt sind.

Analog dazu gibt es auch zwei NetBSD-Versionen. NetBSD/sparc läuft auf älteren 32-bittigen CPUs; das erste offizielle Release war 1994 als NetBSD 1.0 verfügbar. Das ausschließlich auf 64Bit-CPUs lauffähige NetBSD/sparc64 hingegen erschien erst im Jahr 2000 bei der Freigabe von NetBSD 1.5. Abgesehen von diesem Unterschied sind beide NetBSD-Versionen gleich und enthalten auch die gleichen Programme und Pakete.

Erstaunlich ist, daß zusätzlich zwei offizielle NetBSD-Portierungen auf die recht betagten Sun-Maschinen vom Typ Sun 2 und Sun 3 existieren, darunter auch die VME-Modelle mit ihren charakteristischen Einschubkarten. Allerdings basieren diese Oldtimer auf Motorola-CPUs, haben also trotz des Herstellers Sun recht wenig mit Sparc-Systemen gemeinsam.

Das Besondere an Sparc-Systemen ist, daß sie in der Regel statt mit Monitor und Tastatur auch über eine serielle Terminalverbindung bedient werden können, wobei sogar auf das BootPROM (sozusagen das Gegenstück zum BIOS der PC-Welt) zugegriffen werden kann. Dazu wird ein sogenanntes Nullmodemkabel benötigt, das an einem Ende über eine 9-polige SUB-D-Buchse verfügt, damit es zum Beispiel in den COM-Port eines PCs paßt. Je nach Alter und Typ der Sparc-Maschine muß sich auf der Gegenseite solch eines Verbindungskabel ein entsprechender Stecker befinden. Bei sehr alten Rechnern wie einer IPX oder IPC von Sun wird ein 8-poliger Mini-DIN-Stecker benötigt. Da die Computer von Apple früher die gleichen Typen verwendet haben, findet man solch ein Kabel in der Regel beim Zubehörbedarf für Macintosh. Neuere UltraSparc-Systeme besitzen dagegen meist einen üblichen 9-poligen seriellen Anschluß; ein herkömmliches Nullmodemkabel reicht dann aus. Die allermeisten Sparc-Rechner weisen allerdings eine 25-polige D-SUB-Buchse auf, die wie der zweite COM-Port bei AT-PCs beschaltet ist; entsprechende Adapter sind daher leicht zu bekommen.

Die Verbindungseinstellungen für das Terminal sind für die allermeisten Systeme gleich: Die Übertragungsgeschwindigkeit sollte auf 9600 Bits/Sekunde eingestellt sein, es sollten 8 Datenbits, kein Parity und ein Stoppbit verwenden werden. Die geläufige Kurzform für diese Einstellung lautet 9600 8N1, wie sie auch vielen Terminalprogrammen wie zum Beispiel *minicom* findet. Zu beachten ist allerdings, daß eine Sun nur dann Ein- und Ausgabe auf den seriellen Anschluß umleitet, wenn keine Tastatur angeschlossen ist. Außerdem muß das Nullmodemkabel immer am ersten seriellen Port angeschlossen werden, der zweite, mit B gekennzeichnete Anschluß wird nicht angesteuert.

Die Installation von NetBSD kann auf vielfältige Weise geschehen: neben CD-ROMs als Quellmedium wird auch die Installation über ein Netzwerk, sprich von einem NFS- oder FTP-Server unterstützt. Eine Netzwerk-Installation ist gerade beim gleichzeitigen Einrichten von mehreren Rechnern und/oder Workstations ohne Floppy- und CD-

ROM-Laufwerk (wie das zum Beispiel bei einigen Sun Sparcstations der Fall ist) nützlich. Eine Installation ausschließlich mit Disketten wird nicht mehr angeboten; immerhin kann man mittels Floppy booten und dann von einer CD oder einem Server installieren. Vor der Installation von NetBSD 1.6 steht in der Regel allerdings der Download mindestens eines fast 90 Mbyte großen CD-Images. Ein solches ISO-Image enthält alle Pakete, die zu einem grundlegenden System gehören und ist darüber hinaus auch bootfähig. NetBSD findet man auf zahlreichen FTP-Mirrors, eine aktuelle Übersicht findet man auf *http://www.de.netbsd.org/mirrors/*. Hat man einen geeigneten FTP-Server gefunden, wechselt man dort ins NetBSD-Verzeichnis, in der Regel »/pub/NetBSD«. Aus dem Unterverzeichnis »iso/1.6« lädt man entweder das CD-Image »sparccd.iso« für 32-Bit-Systeme oder »sparc64cd.iso« für UltraSparc- beziehungsweise SPARC64-CPUs herunter. Wer den Download scheut und vor allem auch die Packages Collection, also die Sammlung der mehr als 2500 nach NetBSD portierten Programmen und Quelltexten auf CD vorrätig haben möchte, kann auf *http://www.de.netbsd.org/Sites/cdroms.html* aus diversen nationalen und internationalen Anbietern von NetBSD-Bundles wählen. Zwingend notwendig ist dies aber nicht, da man nach Installation des minimalen NetBSDs sämtliche Pakete aus der Packages Collection manuell sowohl von CD-ROMs als auch von FTP-Servern installieren kann.

Wer ohnehin NetBSD über ein Netzwerk installieren will, kann sich einen Rohling sparen und das CD-Image zum Beispiel unter Linux über das Loop-Device einbinden und so die Pakete in den Verzeichnisbaum des NFS- oder FTP-Servers kopieren. Ansonsten sollte jedes Brennprogramm in der Lage sein, aus einem Abbild eine nutzbare CD zu erzeugen.

Grundlegend ist der Umgang mit dem BootPROM, ganz gleich, für welche Installationsart man sich letztendlich entscheidet. Dieser Baustein ähnelt dem BIOS eines PCs, bietet jedoch weitaus mehr Funktionen wie einen Taschenrechner oder eben das Booten über Netzwerk. In dieses Super-BIOS gelangt man, wenn man ein schon installiertes und laufendes Betriebssystem herunterfährt oder wenn man [Stop]+[A] auf der Tastatur drückt beziehungsweise ein Break über die serielle Schnittstelle sendet. Letzteres sollte man allerdings bei einem laufenden Betriebssystem vermeiden, sofern man noch Wert auf dessen Dateisystem legt. Ältere BootPROM-Versionen erkannt man am »>«-Eingabeprompt; durch die Eingabe eines kleinen [n] wechselt man in den »new command mode«, der bei neueren BootPROM-Ausgaben Standard ist und an einem *ok* als Promptzeichen erkennbar ist. Von hier aus hat man verschiedene Möglichkeiten:

Der Befehl *reset* führt einen Reboot aus, während *power-off* den Rechner ausschaltet. Über *printenv* kann man sich diverse Einstellungen ausgeben lassen und mittels *setenv* verändern. So wird zum Beispiel über *setenv ttya-mode 19200,8,n,1,-* die Geschwindigkeit des ersten seriellen Ports erhöht. Zu allen Befehlen gibt es eine brauchbare Hilfe, *help* bietet dafür den geeigneten Einstiegspunkt. Von besonderem Interesse ist das *boot*-Kommando. Normalerweise lädt eine Sparc von der ersten Festplatte ohne jegliches Zutun das Betriebssystem. Fehlt es, versucht sie per Reverse ARP und anschließend per TFTP über einen Netzwerkserver zu booten. Am Prompt des BootPROMs kann man dieses Defaultverhalten umgehen und beispielsweise per *boot cdrom* oder *boot floppy* von anderen Medien booten. Natürlich kann man mittels *boot disk* und *boot net* auch manuell von der Festplatte oder vom Netzwerk booten.

Natürlich müssen einige Vorbereitungen getroffen werden, um NetBSD per Netzwerk zu installieren. Auf jeden Fall wird ein zweiter Rechner im gleichen Netzwerksegment benötigt, um als Server dienen zu können. Auf diesem müssen mindestens ein RARP-, ein TFTP-, ein BOOTPARAM-Daemon sowie ein NFS-Server gestartet sein. Der Netzwerkbootprozeß läuft nämlich folgendermaßen ab: Zunächst versucht das Boot-PROM, mittels Reverse Address Resolution herauszufinden, ob der RARP-Server der MAC-Adresse der Sun eine IP-Adresse zuordnen kann. Ist dies der Fall, lädt das Boot-PROM per Trivial File Transfer Protocol (TFTP) vom RARP-Server einen kleinen Bootloader, der wiederum nach einem Bootparam-Server sucht. Dieser sollte dem Bootloader mitteilen, von welchem Fileserver ein NetBSD-Kernel geladen werden kann. Bei NetBSD/sparc64 enthält dieser Kernel auch eine RAM-Disk samt Installationsprogramm.

NetBSD/sparc bietet solchen Komfort leider nicht, hier muß man zusätzlich ein Miniatur-BSD auf den NFS-Server kopieren. Außerdem muß entweder der NFS- oder ein FTP-Server sämtliche NetBSD-Programmpakete bereithalten.

Damit dieses Zusammenspiel klappt, sollte zunächst die MAC-Adresse beziehungsweise die mit dieser Ethernet-Adresse identische Host-ID der Sun bekannt sein. Beide Angaben werden nach dem Einschalten oder nach einem Reset zusammen mit einigen Informationen über die Sparc-Maschine angezeigt. Auffällig ist, daß die MAC-Adressen nicht von den eingebauten Netzwerkkarten abhängen, sondern nur aus der Host-ID gebildet werden. Tatsächlich vergibt eine Sun allen Netzwerkadaptern die gleiche Hardwareadresse, was normalerweise kein Problem darstellt, da selten mehrere Adapter eines Rechners im selben Netzwerksegment hängen. Wenn allerdings diese Adresse den Wert *ff:ff:ff:ff:ff:ff* besitzt, ist das ein Zeichen dafür, daß das NVRAM, also der batteriegepufferte Baustein, in dem das OpenPROM liegt, defekt oder altersschwach ist und daher ausgetauscht werden sollte.

Treten keine derartigen Probleme auf, ordnet man der MAC-Adresse eine freie IP zu und macht beide dem RARP-Server bekannt. Bei unix-artigen Betriebssystemen fügt man daher in aller Regel der Datei »/etc/ethers« einen neuen Eintrag hinzu:

```
# MAC-Adresse      zugewiesene IP
08:00:20:20:df:e1 192.168.0.2
```

Zusätzlich ist es notwendig, daß die zugewiesene IP-Adresse auch über einen Hostnamen aufgelöst werden kann. Am einfachsten geht das natürlich über eine entsprechende Erweiterung der Datei »/etc/hosts«:

```
# IP-Adresse  Hostname
192.168.0.2   bsd-client
```

Als nächstes kopiert man den Bootloader in das Verzeichnis des TFTP-Servers, meistens nach »/tftpboot«. Bei NetBSD/sparc heißt der Bootloader *boot.net* und liegt auf der CD im Verzeichnis »sparc/installation/netboot«; bei NetBSD/sparc64 hingegen ist es die Datei »ofwboot.net« im Verzeichnis »sparc64/installation/netboot«. Woher weiß nun

aber das BootPROM, welche Datei es vom TFTP-Server laden muß? Es verlangt nach einer Datei, deren Name der hexadezimalen Schreibweise der zugewiesenen IP-Adresse entspricht. Im Beispiel würde eine UltraSparc-Workstation also versuchen, eine Datei namens »C0A80002« zu laden. 32-bittige Suns haben die Eigenart, an diesen Dateinamen noch die Architekturbezeichnung anzuhängen, so daß Workstations wie eine IPC oder eine SparcStation den Bootloader unter dem Namen »C0A80002.SUN4C« beziehungsweise »C0A80002.SUN4M« anfordern. Sinnvollerweise legt man daher unterhalb von »/tftpboot« entsprechende symbolische Links auf »boot.net« oder »ofwboot.net«.

Ist der Bootloader geladen, sendet er Broadcasts aus und wartet auf eine Antwort vom Bootparam-Server, der ihm die IP-Adresse des NFS-Servers und das Verzeichnis verrät, in dem das Kernel-Image zu finden ist. Die Konfiguration hierfür geschieht über die Datei »/etc/bootparams«, deren Aufbau recht simpel ist: durch ein Leerzeichen getrennt gibt man Hostnamen und die Lage des NFS-Verzeichnisses an:

```
bsd-client root=192.168.0.1:/export/install/netbsd
```

Alternativ könnte statt des Bootparam- auch ein dhcp- oder bootp-Server verwendet werden, deren Konfiguration ist allerdings aufwendiger. Zudem wird die eigentliche Funktion eines dhcp-Servers – die dynamische Verteilung von IP-Adressen – hier nicht benötigt.

Abschließend müssen noch der NFS-Server beziehungsweise die von ihm exportierten Verzeichnisse eingerichtet werden. Auf jeden Fall muß das Verzeichnis »/export/install/netbsd« exportiert werden. Für NFS-Server unter BSD und Linux muß hierzu ein entsprechender Eintrag in der Datei »/etc/exports« vorhanden sein, bei Solaris hingegen in »/etc/dfs/dfstab«. Vorzugsweise sollte das Verzeichnis schreibgeschützt exportiert werden. Für NetBSD/sparc64 reicht es, das Kernel-Image »netbsd.INSTALL.gz« von der CD-ROM aus dem Verzeichnis »sparc64/binary/kernel« nach »/export/install/netbsd« zu kopieren. Zusätzlich muß dort ein symbolischer Link mit dem Namen *netbsd* auf das Kernel-Image zeigen. NetBSD/sparc bringt leider kein Image samt RAM-Disk mit. Daher kopiert man zunächst die Datei »sparc/binary/kernel/netbsd-GENERIC.gz« von der CD auf die Festplatte und legt auch hier einen symbolischen Link unter dem Namen *netbsd* an. Anschließend muß man im Verzeichnis »/export/install/netbsd« das Archiv »rootfs.tgz«, das auf der CD in »sparc/installation/netboot« liegt, entpacken. Außerdem muß bei beiden Architekturen das Verzeichnis »sparc64/binary/sets« beziehungsweise »sparc/binary/sets« nach »/export/install/netbsd« kopiert werden. In Kurzform bedeutet dies für NetBSD/sparc64 (unter Annahme, daß die CD nach »/cdrom« gemountet ist):

```
$ cd /export/install/netbsd
$ cp /cdrom/sparc64/binary/kernel/netbsd.INSTALL.gz .
$ ln -s netbsd.INSTALL.gz netbsd
$ cp -r /cdrom/sparc64/binary/sets .
```

Analog für NetBSD/sparc:

```
$ cd /export/install/netbsd
$ cp /cdrom/sparc/binary/kernel/netbsd-GENERIC.gz .
$ ln -s netbsd-GENERIC.gz netbsd
$ cp /cdrom/sparc/installation/netboot/rootfs.tgz .
$ tar -xzf rootfs.tgz
$ cp -r /cdrom/sparc/binary/sets .
```

Die neu konfigurierten Serverdienste sollten natürlich nun noch (neu) gestartet werden. Bei korrekter Netzwerkverkabelung reicht jetzt die Eingabe von *boot net* auf der Boot-PROM-Konsole, damit die Sparc-Maschine über das Netzwerk bootet.

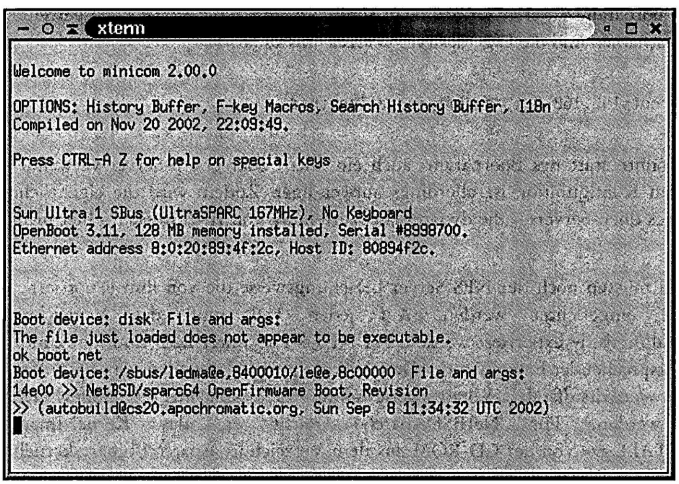


Abb. 2.40: Eine Sun bootet übers Netzwerk

Möchte man nur eine einzige Workstation mit NetBSD aufsetzen und ist diese ferner mit einem CD-Laufwerk ausgestattet, bietet sich die Installation per bootfähiger CD an. Dazu gibt man *boot cdrom* an der PROM-Konsole ein, nachdem man den normalen Startvorgang der Sparc-Maschine mittels [Stop]+[A] auf der Tastatur oder durch Senden eines Breaks auf dem seriellen Terminal unterbrochen hat. Die Installation von NetBSD/sparc kann ferner auch über Bootdisketten erfolgen; der entsprechende Boot-PROM-Befehl lautet *boot floppy*. Die beiden Disketten-Images findet man im Verzeichnis »sparc/installation/floppy« auf der CD. Unter Unix kann man *dd* verwenden, um diese Abbilder auf zwei Floppies zu übertragen. Windows-Benutzer können für diesen Zweck das Programm *rawrite* verwenden. Allerdings liegen die einzelnen NetBSD-Pakete nicht als Disketten-Image vor, so daß dennoch auf einen Server oder die CD-ROM als Installationsquelle zurückgegriffen werden muß.

Nach dem Booten der Sparc bestätigt man die Frage nach der Art des Terminals mit einem [Return]. NetBSD/sparc bietet danach noch die Wahl zwischen den alten Instal-

lationsskripten und dem neuen, menügeführten *sysinst* an, das deutlich mehr Flexibilität bietet und das durch Eingabe eines [I] und Bestätigung mittels [Return] gestartet wird. NetBSD/sparc64 hingegen startet automatisch *sysinst*, so daß man gleich freundlich begrüßt wird:

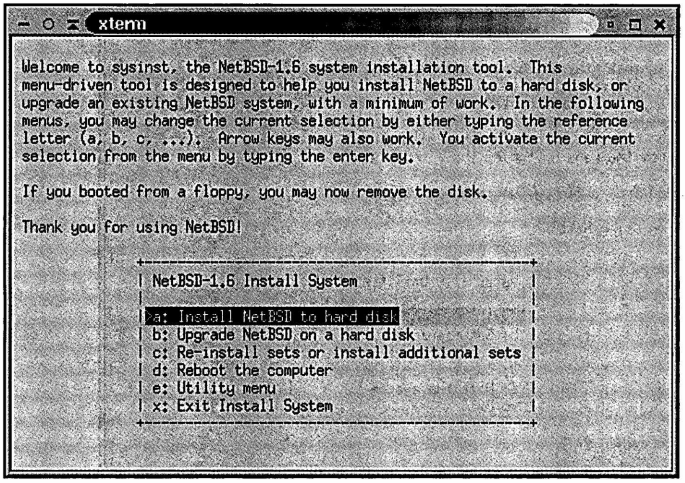


Abb. 2.41: Hauptmenü

Die Bedienung des Menüs erfolgt entweder mit den Cursortasten oder durch Eingabe des Buchstabens, der vor einer gewünschten Option steht, gefolgt von einem [Return]. Wer die Sparc über Tastatur und Monitor bedient, muß mit dem amerikanischen Keyboard-Layout vorlieb nehmen. Die Punkte *b* und *c* des Hauptmenüs sind relevant, wenn schon ein NetBSD installiert wurde und dieses nun aktualisiert oder ergänzt werden soll. Möglichkeit *d* rebootet die Sparc-Maschine, während Option *x* das Installationsprogramm beendet und man sich anschließend an einer Shell wiederfindet. Punkt *e* ruft ein Hilfsmenü auf (Abb. 2.42). Hier kann man vor allem die Netzwerkparameter nachträglich konfigurieren.

Wer zum Beispiel die NetBSD-Pakete nicht vom lokalen NFS-Server installieren möchte, sondern direkt von einem FTP-Server im Internet, kann unter Punkt *c* des Hilfsmenüs IP-Adressen für den Default-Router und Nameserver angeben. Verfügt die Sparc über mehrere Netzwerkinterfaces, kann man natürlich dem Installationsprogramm mitteilen, welches verwendet werden soll. Über die Shell, die über Option *a* gestartet wird, freut man sich besonders, wenn ein schon installiertes NetBSD-System nicht mehr bootet – sei es durch Fehlkonfiguration oder einen Defekt der Hardware.

In den meisten Fällen wird man auf diese Hilfen nicht angewiesen sein und lediglich Punkt *a* im Hauptmenü auswählen, um als erstes die Festplatte(n) zu partitionieren. Befinden sich noch wichtige Daten auf den Festplatten, sollten Sie die Sicherheitsabfrage mit »Nein« beantworten und ein Backup anlegen. Ansonsten wählt man die ge-

wünschte Festplatte aus, auf der NetBSD installiert werden soll. Verfügt das System über nur eine Festplatte, entfällt diese Auswahl logischerweise. Die anschließende Partitionierung kann auf vier Arten geschehen: Zum einen kann *sysinst* selbst den erforderlichen Platz für eine NetBSD-Installation mit oder ohne X-Window-Systeme ermitteln.

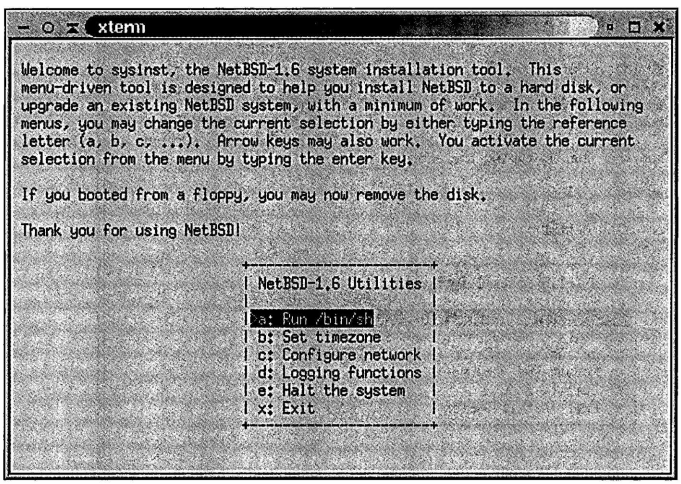


Abb. 2.42: Hilfsmenü

Andererseits kann die Festplatte von Anfang an nach eigenen Wünsche aufgeteilt werden oder die vorhandene Partitionierung übernommen werden. Letzteres sollte vor allem gewählt werden, wenn man NetBSD parallel zu einem anderen Betriebssystem installieren möchte. Unabhängig von dieser Auswahl kann die Partionierung vor dem Kopieren der Programmpakete noch manuell angepaßt werden. Solange also kein installiertes System gerettet und übernommen werden soll, sind die beiden erstgenannten Möglichkeiten ein guter Ausgangspunkt.

Wählt man Möglichkeit *a* oder *b* im Partionierungsmenü, muß noch entschieden werden, welche Unterverzeichnisse auf einer eigenen Partition liegen sollen. Das Root-Verzeichnis (»/«) ist wie bei jedem Unix-System obligatorisch, ebenso sollte ein eigener Swap-Bereich angelegt werden. Bei sehr kleinen Festplatten sollte hingegen auf eine separate Partition für das »/usr«-Verzeichnis verzichtet werden, um so den geringen Speicherplatz möglichst gut auszunutzen. Ferner kann es sinnvoll sein, das »/tmp«-Verzeichnis so auszulagern, daß es einen bestimmten Teil des Arbeitsspeichers erhält. NetBSD realisiert dies über das *Memory Filesystem* (mfs): Das so gemountete Verzeichnis und die darin enthaltenen Dateien werden nicht auf einer Festplatte abgelegt, sondern werden bis zum nächsten Reboot oder Unmount im Arbeitsspeicher gehalten. Damit kann unter Umständen ein Überlaufen des Root-Verzeichnisses verhindert werden. Ob ein Auslagern der beiden Verzeichnisse »/var« und »/home« angebracht ist, hängt vom Einsatzzweck des Rechners ab. Unterhalb von »/var« werden hauptsächlich Logfiles und

temporäre Daten wie aufbereitete Druckdaten gespeichert. Steht ein Betrieb als Server im Vordergrund, sollte »/var« seine eigene Partition erhalten. Gleiches gilt für das »/home«-Verzeichnis: Sollen später viele Benutzer eingerichtet werden und jeder dieser Benutzer auch sein eigenes Homeverzeichnis besitzen, ist es empfehlenswert, auch hierfür eine eigene Partition anzulegen.

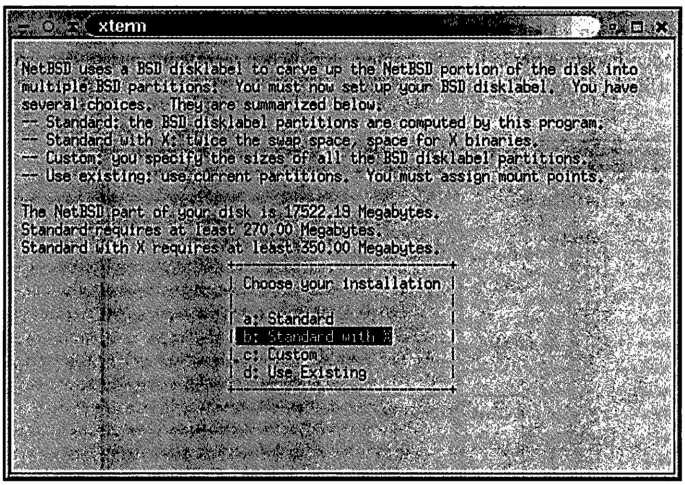


Abb. 2.43: Partionierungsmenü

Leider kann *sysinst* noch nicht mehrere Festplatten einbinden, so daß einzelne Verzeichnisse nicht schon während der Installation auf anderen Laufwerken liegen können. Sollen sich zum Beispiel die Homeverzeichnisse aus Geschwindigkeits- und Platzgründen auf einer eigenen Festplatte befinden, legt man bei der Installation zunächst keine eigene Partition dafür an. Nach dem ersten eigenständigen Booten des NetBSD-Systems partitioniert und formatiert man die gewünschte Festplatte mountet sie dann nach »/home« – möglichst noch bevor ein Benutzer angelegt wird.

Sind diese Vorüberlegungen abgeschlossen, verläßt man das Filesystem-Menü über den Punkt *Exit*:

Wählt man hingegen im Partitionierungsmenü den Punkt *c*, fragt das Installationsprogramm der Reihe nach die gewünschte Größe für das Root-Verzeichnis, die Swap-Partition, die Verzeichnisse »/usr«, »/var« und »/home« ab. Zusätzlich können dann noch weiteren, frei wählbaren Verzeichnissen eigene Partitionen zugeordnet werden, solange noch unpartitionierter Speicherplatz zur Verfügung stellt und noch nicht mehr als sieben Partitionen erstellt wurden.

Danach besteht noch die Möglichkeit, von Hand die Festplattenaufteilung vorzunehmen. Wer im Partitionierungsmenü die Option *d* wählt, landet sofort und ohne Hilfestellung an dieser Stelle, dem Disklabel-Menü. Pro (logischer) Festplatte, sprich: pro

Disklabel kann NetBSD/sparc maximal acht Partitionen anlegen, fortlaufend von *a* bis *h* bezeichnet, wobei allerdings Partition *c* stellvertretend für die gesamte Festplatte steht und daher nicht über ein Verzeichnis gemountet werden kann. Eine ungeschriebene Regel verlangt ferner, daß auf Partition *a* das Root-Verzeichnis und auf Partition *b* der Swap-Bereich liegen, was natürlich nicht eingehalten werden kann, wenn zum Beispiel ein bereits installiertes Solaris erhalten bleiben soll. Das manuelle Editieren der Festplattenaufteilung ist allerdings ein wenig umständlich, da der Offset, also der Startpunkt einer Partition nicht automatisch verschoben wird, wenn man die Größe der davor liegenden Partition verändert. So können dazwischen leere, ungenutzte Bereiche entstehen – Überlappungen fängt *sysinst* hingegen ab.

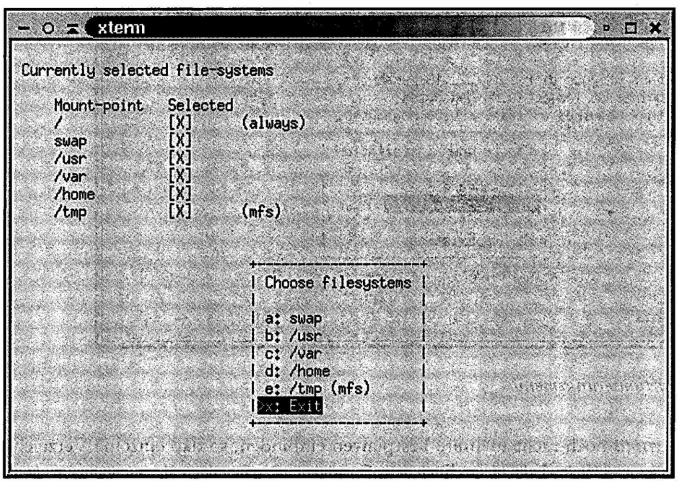


Abb. 2.44: Dateisystem-Menü

Möchte man zum Beispiel den Speicherplatz unterhalb von »/usr« auf Kosten der Homeverzeichnisse erhöhen, wählt man den Punkt *a - Change partitions*«. Hier wählt beziehungsweise editiert man zunächst Partition *g*, auf der später »/usr« gemountet wird. Eine neues Untermenü fragt ab, welche Eigenschaft der Partition verändert werden soll. Über den zweiten Punkt können Startpunkt und Größe gesetzt werden. Als Offset ist der aktuelle Startpunkt der Partition (im Beispiel bei Zylinder 606) zu übernehmen. Danach gibt man die gewünschte Größe ein – also »2048M« für 2 Gbyte. Nach dem Verlassen dieses Untermenüs ist zu sehen, daß sich Partition *g* und *e* überlappen: Der Startzylinder von *e* liegt vor dem Ende von Partition *g*, wie in Abbildung 2.46 dargestellt.

Um dies zu beheben, muß der Offset der Partition *e* erhöht werden. Also ruft man das Untermenü zu Punkt *e* auf und wählt erneut *Offset/size*. Bei der Frage nach dem Offset genügt hier die Eingabe von »g«: Das Installationsprogramm weiß dann, daß die Partition *e* genau hinter *g* liegen soll.

Auch bei der Größe muß man sich keine Zylinderangaben merken: Da die Homeverzeichnis-Partition am Ende des Festplatte liegt, muß nur »-1« angegeben werden, um sie auf maximale Größe auszudehnen.

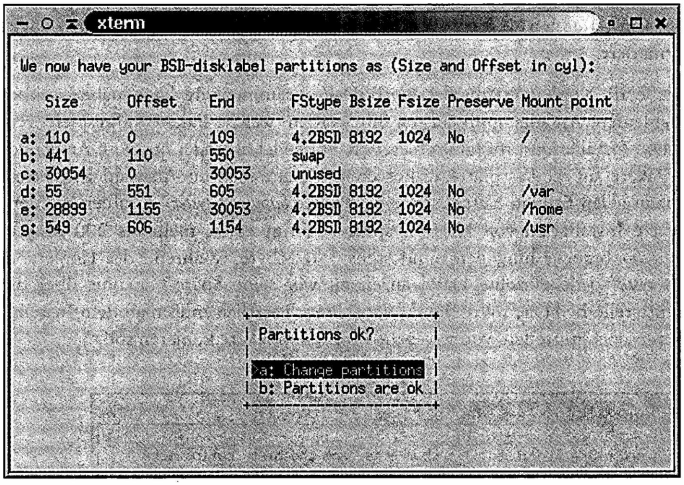


Abb. 2.45: »/usr« ist zu klein

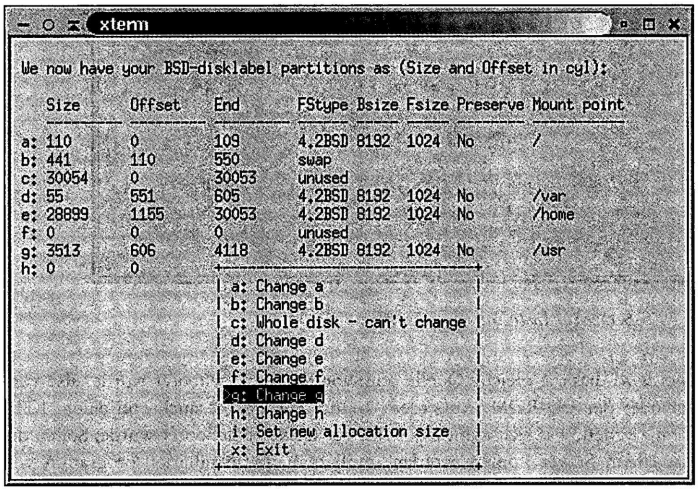


Abb. 2.46: Partition g und e überlappen einander

Ist die Aufteilung der Festplatte abgeschlossen, müssen Sie im Disklabel-Menü den Punkt *b* auswählen. Danach muß noch ein Name für die Festplatte vergeben werden. Da dieser eher kosmetischer Natur ist, kann durchaus die Vorgabe »mydisk« verwendet werden. Eine letzte Sicherheitsabfrage warnt davor, daß nun der Inhalt der Festplatte verändert wird. Beantwortet man mit »No«, kehrt man zum Hauptmenü zurück und alle Partitionseinstellungen werden verworfen. Ansonsten werden die Partionen angelegt und formatiert.

Der abschließende Kopiervorgang der NetBSD-Programmpakete kann nahezu unbeaufsichtigt ablaufen. Anfangs muß ausgewählt werden, welche Pakete aufgespielt werden sollen. Das Installationsprogramm bietet an, entweder alle oder nur vom Anwender bestimmte Pakete auf die Festplatte zu kopieren. Möchte man die Auswahl selbst treffen, müssen auf jeden Fall ein Kernel sowie die Pakete *base* und *system* installiert werden; eine komplette Installation von NetBSD/sparc benötigt etwas mehr als 200 Mbyte, NetBSD/sparc64 braucht hingegen mindestens 350 Mbyte. Während des Kopiervorgangs kann *sysinst* jede einzelne Datei anzeigen; wer seine Sparc-Maschine über die serielle Schnittstelle bedient, sollte darauf verzichten. Letztlich muß man dem Installationsprogramm noch mitteilen, von wo es die Programmpakete kopieren soll.

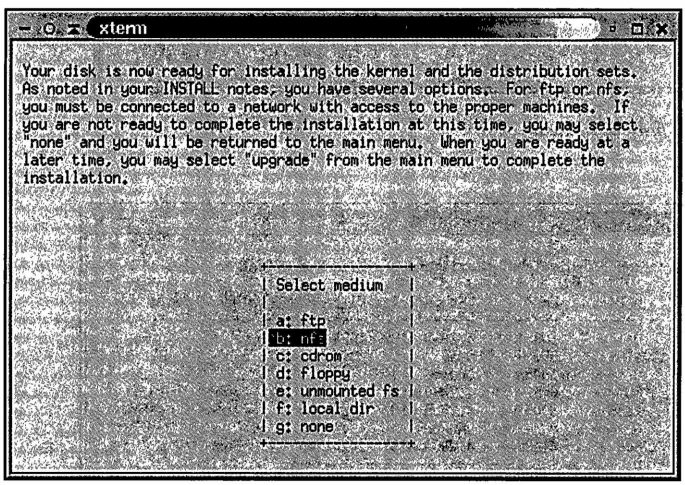


Abb. 2.47: NFS-Server zur Installation auswählen

In der Regel wird dies die gleiche Quelle sein, von der auch gebootet wurde, also vom NFS-Server oder der CD-ROM. Außerdem können die Pakete auch von einem FTP-Server geholt werden, was sich anbietet, wenn per Diskette gebootet wurde. Sollen die Pakete von einem Server kopiert werden, muß man im Installationsprogramm von NetBSD/sparc64 zumindest die IP-Adresse der Sparc-Maschine angeben; NetBSD/sparc hingegen kann die Netzwerkkonfiguration übernehmen, die es vom Bootparam-Server zugeteilt bekommen hat. Auf jeden Fall müssen Sie die IP-Adresse des NFS- oder FTP-

Servers angeben sowie das Verzeichnis, in dem die NetBSD-Pakete auf dem Server liegen.

Nun kann der Kopiervorgang beginnen, der je nach Geschwindigkeit der eingesetzten Systeme und Anzahl der ausgewählten Pakete bis zu 15 Minuten dauert.

Abschließend muß das NetBSD-System noch grundlegend konfiguriert werden. Hat man die aktuelle Netzwerkkonfiguration, die zur Installation notwendig war, wohl überlegt geplant und möchte man daher diese Werte übernehmen, kann *sysinst* diese in den Konfigurationsdateien unterhalb von »/etc« auf der Festplatte speichern. Wie andere Unix-Derivate verlangt auch NetBSD nach einer Zeitzone, mit der die lokale Ortszeit bestimmt wird; hierbei können natürlich auch geographische Regionen wie »Europe/Berlin« gewählt werden. Die Bedienung des Zeitzonenmenüs ist etwas gewöhnungsbedürftig: Man wählt mit den Cursor- beziehungsweise den PageUp/PageDown-Tasten die entsprechende Region aus und übernimmt die Auswahl mittels [Return]. Zum Verlassen ist ein kleines [x], ebenfalls gefolgt von einem [Return] einzugeben. Anschließend muß die Art, mit der die Benutzerpaßwörter verschlüsselt werden sollen, festgelegt werden; hier sollte man auf jeden Fall die Verwendung von MD5-Hashes der DES-Verschlüsselung vorziehen.

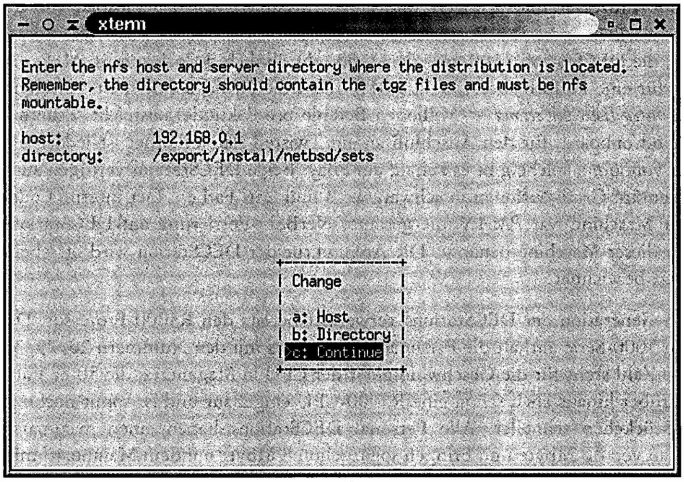


Abb. 2.48: NFS-Pfad angeben

Damit das frische NetBSD-System nach dem ersten Reboot nicht gleich mißbraucht wird, müssen Sie im letzten Schritt ein *root*-Passwort vergeben, das sorgfältig festgelegt werden sollte. Als Faustregel für Paßwörter gilt: Anfangs- oder Endbuchstaben in zufälliger Groß- und Kleinschreibung aus den Wörten eines Liedrefrains zusammenstellen, wobei einzelne Buchstaben durch Sonderzeichen zu ersetzen oder zu ergänzen sind, zusätzlich versehen mit einigen Ziffern.

Danach ist die Installation von NetBSD abgeschlossen, man befindet sich wieder im Hauptmenü. Punkt *d* rebootet die Sparc, CD-ROMs und Disketten sollten zuvor aus den Laufwerken entfernt werden. Nun wird das auf der Festplatte installierte BSD-System hochgefahren.

2.5 Die Installation auf der DECStation

von Kevin Read

Die Firma Digital Equipment Corporation (DEC) ist für seine VAX- und Alpha-Systeme bekannt und für das Betriebssystem VMS. Als Mitte der achtziger Jahre RISC-Prozessoren[1] Mode wurden, wurde DEC-intern zuerst an einem eigenen Prozessor namens PRISM gearbeitet, die Arbeiten wurden aber eingestellt. Die Erkenntnisse flossen jedoch später in Alpha ein. Also mußte ein RISC-Prozessor einer anderen Chipschmiede herhalten. Man entschied sich für die R-Serie der Firma MIPS, die unter anderem auch von Silicon Graphics verbaut wurden. Die Maschinen waren von Anfang an dafür ausgelegt, unter der DEC-Spielart von Unix namens *Ultrix* zu operieren, es gibt keine Portierung von VMS.

Der erste Sproß der DECStation-Familie war die DECStation 2100. Ausgestattet mit einem R2000-Prozessor mit 12 MHz und einem maximalen Speicherausbau von 24 Mbyte, war sie dem Intel-386-Prozessor zu dessen Blütezeit weit voraus. Der direkte Nachfolger, die DECStation 3100, war der 2100 sehr ähnlich, jedoch mit 16 MHz getaktet, was für eine Performance von 11 MIPS (nicht die Firma, sondern ein Akronym für *Million instructions per second* = Millionen Befehle pro Sekunde) ausreicht. Ethernet und SCSI sind onboard, für den Anschluß an ein Twisted-Pair-Ethernet ist jedoch ein Transceiver vonnöten, solche gibt es zuhauf auf eBay. Beide DECStations verfügen auch über rudimentäre Grafikoptionen in schwarz/weiß und 256 Farben. Der interne Codename dieser Maschine war *PMAX*, die gesamte NetBSD-Portierung auf DECStations wurde nach dieser Maschine benannt. Die Architektur der DECStation wird oft ebenfalls mit *pmax* bezeichnet.

Die nächste Generation der DECStations verwendet bereits den R3000-Prozessor. Die DECStation 5000-Serie enthält die Personal DECStations mit den Nummern 20, 25, 33 und 50. Die Zahl steht für die Geschwindigkeit der CPU in Megahertz, die 50-er Variante hat darüber hinaus auch schon eine R4000-CPU eingebaut und ist somit nochmal ein gutes Stückchen schneller. Alle Personal DECStations haben einen maximalen RAM-Ausbau von 40 Mbyte und bringen sogar schon 8 Mbyte auf dem Mainboard mit. Es gibt zwei Turbochannel-Steckplätze für Erweiterungen, die üblichen Workstation-Anschlüsse wie SCSI und Ethernet sind auch dabei. Als nächste Mitglieder der DECStation 5000-Serie gibt es die 120, 125, 133 und 150. Wie bei der Personal DS auch, geben hier die letzten beiden Ziffern des Modells Aufschluß über die Geschwindigkeit des Prozessors, wobei die 150 wieder über einen R4000 verfügt. Die Systeme lassen sich über drei Turbochannel-Steckplätze gut aufrüsten, es gibt auch verschiedene Grafikkarten mit teilweise sehr hoher Bildqualität. Mit maximal 128 Mbyte RAM kann man den

[1] Reduced Instruction Set CPU, CPUs mit wenigen einfachen, aber schnellen Befehlen

Geräten auch speicherfressende Applikationen zumuten. Die letzten drei Mitglieder der 5000-Serie sind die 200, 240 und 260. Während in der DECStation 5000/200 ein eher lahmer 25 MHz R3000 steckt, beeindrucken die 240 mit einem 40 MHz R3000 und die 260 mit einem rasanten 60 MHz R4400, die schnellste aller DECStations. Auch die 200-er besitzen drei Turbochannel-Stecker, die jedoch im Gegensatz zur 100-er mit 25 statt 12,5 MHz getaktet sind. Der maximale Speicherausbau von 480 Mbyte genügt auch aktuellen Anwendungen. Bei den 100- und 200-Modellen befinden sich die CPUs auf eigenen kleinen Karten und lassen sich somit gut gegen schnellere austauschen, jedoch nur innerhalb der Serie. Alle 5000er außer der 5000/200 haben nur einen AUI-Ethernet-Anschluß, benötigen also einen Transceiver, um mit Twisted-Pair oder BNC-Netzwerken sprechen zu können.

Da NetBSD seit Version 1.5 schon einiges an Speicher benötigt, ist ein Betrieb mit weniger als 12 Mbyte RAM nur für Kernel-Entwickler zur Speicheroptimierung des Systems sinnvoll. Generell wird der Einsatz von DECStation 2100 und 3100 unter NetBSD keine große Arbeitsfreude aufkommen lassen, die 5000-er Serie ist für ein solch modernes Betriebssystem wesentlich besser geeignet. Ich beziehe mich im folgenden primär auf diese Modelle, Abweichung zur 2100/3100 werden aufgeführt.

Als Arbeitsgerät dient eine DECStation 5100/133 aus der mittleren Serie mit einem 33 Mhz R3000-Prozessor. Von den drei Turbochannel-Steckplätzen ist einer mit einer PMAG-B-Grafikkarte bestückt, die anderen sind frei. Eine Übersicht über alle verfügbaren Erweiterungen und über eventuelle Treiber für NetBSD gibt es unter [1]. Wenn aber keine Tastatur oder kein Bildschirm für das verwendete Modell vorhanden ist, ist das bei der PMAX kein Problem. Wie die meisten Workstations läßt sich auch die DECStation über ein serielles Terminal bedienen, etwa einen Windows-PC mit Hyperterm, ein Linux- oder BSD-System mit *minicom* oder ein »echtes« Terminal wie das VT100 oder VT220 von DEC. Wenn das System beim Systemstart keine Tastatur entdeckt, wird automatisch auf die serielle Konsole umgeschaltet. Alternativ kann auch die Grafikkarte ganz entfernt werden, dann wird immer die serielle Konsole verwendet. Dies kann jedoch auch später im Bootmonitor eingestellt werden. Bei DECStations 5100/1xx und 5100/2xx sollte die serielle Konsole an den dritten seriellen Port angeschlossen werden, derjenige, der am weitesten vom Netzteil entfernt ist. Die DECStations/xx gibt die Informationen auf dem DB25-Port aus, also dem großen seriellen Stekker. Die DECStation 2100 und 3100 verwenden die eingebaute Grafikkarte, solange diese nicht ausgebaut und der Mausport mit einem DEC-eigenen Terminator überbrückt wird. Selbst dann muß die serielle Schnittstelle erst von dem seltenen RS423-Format auf das übliche RS232-Format umgewandelt werden. Bei den alten Maschinen steht dem bildschirmlosen Betrieb also einiges im Weg.

Nach dem Anschalten sollte die DECStation erst einmal einen Selbsttest durchführen. All diese Ausgaben sind auf dem Bildschirm oder der seriellen Konsole vollkommen identisch. Das sieht in etwa so aus:

```
KN02-BA V5.7j
3/misc/kbd
?STF (4: Ln#0 Kbd self test)
```

```
3/misc/mouse
?STF (4: Ln#1 Pntr self test)

3/cache/seg
```

Dies sind Meldungen des Bootmonitors. Am ehesten läßt sich dieses fest in jedes System eingebaute Programm mit dem BIOS des PCs vergleichen, es ist jedoch mächtiger. Beim Systemstart wird zuerst ein Selbsttest des Systems und der eingebauten Komponenten gestartet. In obigem Beispiel wurde zuerst die Systemversion ausgegeben, Informationen zu den jeweiligen Versionen gibt es unter [2]. Die nächsten beiden Zeilen deuten darauf hin, daß Keyboard (Kbd) und Maus (Pntr = Pointer) fehlen. Das Gerät ist über die serielle Schnittstelle angeschlossen. Stehen hier noch andere Fehlermeldungen, könnte der Selbsttest einen Systemfehler entdeckt haben. Ein Handbuch für die DS 5000/xx-Serie, das auch über andere 5000-er Modelle Auskunft gibt, gibt es unter [3] zum Download. Es liegt im gezippten Postscript-Format vor – also abspeichern, mit *gunzip* entpacken und dann mit einem Postscript-Viewer wie *ggv* betrachten. Dort stehen auch mehrere interessante Methoden der Fehlerdiagnose, auf die ich hier nicht näher eingehen möchte. Nachdem der Selbsttest abgeschlossen ist, erwartet der Prompt des Bootmonitors die Eingaben mit >>. Steht stattdessen ein R> auf dem Bildschirm, ist das Gerät durch ein Kennwort geschützt. Ist es nicht bekannt, kann es durch Umstecken eines Jumpers zurückgesetzt werden, mehr dazu in den FAQs unter [4]. Durch Eingabe des Fragezeichens erhält man prompt einen Überblick über die möglichen Befehle:

```
>>?
? [cmd]
boot [[-z #] [-n] #/path [ARG...]]
cat SCRPT
cnfg [#]
d [-bhw] [-S #] RNG VAL
e [-bhwcdoux] [-S #] RNG
erl [-c]
go [ADR]
init [#] [-m] [ARG...]
ls [#]
passwd [-c] [-s]
printenv [EVN
restart
script SCRPT
setenv EVN STR
sh [-belvS] [SCRPT] [ARG..]
t [-l] #/STR [ARG..]
unsetenv EVN
```

Man sieht schon an der Menge der Befehle, daß das Boot-ROM der DECStations relativ mächtig ist. Ich möchte hier nur auf vier Befehle näher eingehen. Weitere Informatio-

nen zu den Befehlen insgesamt geben die FAQs zu NetBSD/pmax unter [4] und das Handbuch in [3]. Vor allem bei einem neuen oder unbekannten System kann der Befehl *cnfg* sehr hilfreich sein. Er gibt eine genaue Übersicht über die im System installierten Optionen. Bei meiner DECStation 5000/125, aus der die Grafikkarte zuvor entfernt wurde, gibt der Befehl folgendes aus:

```
3: KN02-BA  DEC  V5.7j  TCF0 ( 64 MB) (enet: 08-00-2b-34-ed-e4) (SCSI = 7)
```

Die erste Zahl ist die Slotnummer. Da die ersten drei Slots mit den Nummern Null bis Zwei keine Karten enthalten, werden sie nicht angezeigt. Slot Nummer Drei ist das System selbst. Es handelt sich um eine KN02-BA-Backplane der Firma DEC (wie zu erwarten war), das Boot-ROM hat Version 5.7j, TCF0 ist die kryptische Angabe des Firmware-Typus. Alle danach folgenden Informationen sind Details zu diesem Slot. In diesem Fall läßt sich erkennen, daß das System 64 Mbyte RAM besitzt, die integrierte Netzwerkkarte obige Ethernet-Adresse besitzt und der Onboard-SCSI-Controller auf die SCSI-ID 7 hört.

Der Befehl *setenv* setzt die Umgebungsvariablen im Boot-ROM, ähnlich wie unter Unix die Shellvariablen der C-Shell. Er nimmt zwei Parameter, zuerst den Namen der Variable, dann den Wert. Als Beispiel setzt der Befehl *setenv haltaction "h"* die Variable *haltaction* auf den Wert *h*. Dies bewirkt, daß die Maschine nach dem Start nicht direkt ins Betriebssystem bootet, sondern im Boot-Monitor stehen bleibt. Der Wert *b* befiehlt dem System dagegen, direkt ins Betriebssystem zu starten. Einen Überblick über alle Systemvariablen samt aktuellen Werten gibt der Befehl *printenv*. Auch hier geben die pmax-FAQ [4] und das DECStation-Handbuch [3] weiter Auskunft.

Der wichtigste Befehl ist wohl *boot*, der das Betriebssystem startet. Interessanterweise nimmt er verschiedene Optionen entgegen, die wir später beim Installieren benötigen werden. Die Syntax des Befehls für die 5000-er Maschinen lautet grundsätzlich *boot {-Optionen} {Pfad} {Argumente}*. Die Optionen spielen meistens keine Rolle und können somit erst einmal außer Acht gelassen werden. Der Pfad bezeichnet das zu startende Betriebssystem und setzt sich aus den Komponenten *#/Gerät/Dateiname* zusammen. Die Raute steht für die Zahl des zu verwendenden Turbochannel-Slots. Als Geräte stehen einerseits die SCSI-Geräte als *rz0* bis *rz6* zur Verfügung, die Zahl steht dabei für die SCSI-ID. Festplatten, CD-ROMs und andere SCSI-Wechselmedien werden dabei mit *rz* bezeichnet, Bandlaufwerke mit *tz*. SCSI-CD-ROMs haben an der DECStation jedoch noch eine Beschränkung, dazu später noch mehr. Weitere Geräte sind *mop* und *tftp* für das Booten übers Netzwerk. Die Angabe eines Dateinamens ermöglicht die Wahl der zu startenden Datei. Getestet habe ich dies nur für SCSI-Geräte, ob es beim Booten übers Netzwerk auch funktioniert, konnte ich nicht testen. Dies ist jedoch kein größeres Hindernis, da sich der Dateiname ja im BOOTP-Server einstellen läßt. Es ist natürlich nicht sinnvoll, die Nummer eines Slots mit Netzwerkkarte für den Systemstart mit einem *rz*-Gerät zu verwenden. Als Argumente können noch Optionen für Singleuser- oder Multiuser-Betrieb an den zu startenden Kernel übergeben werden. Dies alles mag auf den ersten Blick übermäßig kompliziert für eine »einfache« Sache wie das Booten erscheinen, doch der *cnfg*-Befehl gibt alle fürs Booten benötigten Informationen aus. Außerdem kann die Umgebungsvariable *boot* auf den gewünschten Pfad gesetzt werden.

Und schon genügt ein einfaches *boot*, um das so eingestellte Betriebssystem zu starten. Tabelle 1 listet ein paar Beispiele für sinnvolle Bootbefehle auf. Der Bootbefehl der 2100-er und 3100-er DECStations ist ähnlich aufgebaut. Zum Booten übers Netzwerk ist die Syntax *boot -f tftp()*, vom SCSI-Gerät mit der ID *x* ist es *boot -f rz(0,x,0)Dateiname*, statt *Dateiname* muß natürlich der Name einer zu startenden Datei verwendet werden.

Beginnen wir also mit der Installation des NetBSD-1.6-Systems. Für manche Maschinen, die Probleme mit dem Booten großer Kernel-Images aus dem Netzwerk haben, mag die Verwendung von 1.5.3 sinnvoll sein, da dort noch auf die Kernelgröße geachtet wurde. Auf den DECStations 2100 und 3100 sind ältere NetBSD-Versionen sehr viel bequemer. Aktuellere Versionen des Betriebssystems zeigen auf langsameren und speicherarmen Systemen inzwischen ihre Neigung zu »Bloat«, zu großem Codeumfang und der Optimierung für schnelle neue Systeme. Versionen bis 1.4 sind eindeutig besser für solche Maschinen geeignet. Prinzipiell ist der Installationsvorgang auf allen Plattformen gleich. Zuerst muß das Installationssystem auf ein für die DECStation sinnvolles Medium gebracht werden. Gestartet werden kann die Installation von einer Festplatte, einem CD-ROM oder übers Netzwerk. CD-ROM-Laufwerke müssen jedoch für die Zusammenarbeit mit einer DECStation (und an vielen anderen Workstations auch) den Betrieb mit einer Sektorgröße von 512 Bytes beherrschen – PCs verwenden 2048 Byte große Sektoren. Viele Laufwerke unterstützen diesen Modus nicht. Ich konnte nur meinen Plextor-Brenner PX-R820T zur Mitarbeit überreden, andere SCSI-Geräte wollten nicht. Somit fällt die an sich bequemere Installation über CD-ROMs leider häufig aus. Eine gute Alternative wäre das Aufspielen des Boot-Images auf eine unbenutzte SCSI-Platte. Wer die Arbeit mit Ausbauen, Einbau in einen anderen SCSI-fähigen Rechner und erneuten Umbau nicht scheut, kann diesen Weg nehmen. Insgesamt am einfachsten ist meist die Installation über das Netzwerk. Dies setzt nur einen weiteren Rechner im Netzwerk voraus, kompatibel ist jeder PC mit einem geläufigen Unix wie NetBSD, FreeBSD oder Linux, und auch die meisten Unix-Workstations. Auch mit Windows als Server sind schon erfolgreich Workstations installiert worden, mehr dazu unter [10]. Falls möglich, empfehle ich die Installation über das Netzwerk.

2.5.1 Installation über das Netzwerk

Für eine Installation über das Netz benötigen wir vom FTP-Server der NetBSD-Org einen Kernel im ECOFF-Format und ein entsprechendes Installations-Diskimage. Beides kann von *ftp.de.netbsd.org* bezogen werden, dem deutschen Spiegelserver des Projekts. Alternativ bietet sich natürlich auch der Hauptserver in den USA, *ftp.netbsd.org*, an. Im Unterverzeichnis des jeweiligen Releases im »installation«-Verzeichnis befindet sich die Datei »netboot/diskimage.tgz«. Im »binary«-Verzeichnis ist die Datei »kernel/netbsd-INSTALL.ecoff.gz« zu finden. Bei den NetBSD-Versionen vor 1.6 heißt die Datei »nfsnetbsd.gz«. Die gesamten URLs stehen auch unter [5] und [6]. Für andere NetBSD-Versionen lauten die Dateinamen gleich, nur der Pfad ändert sich natürlich entsprechend. Für das Netbooting wird auf dem Server, also dem zweiten im Netzwerk befindlichen Rechner, am besten ein völlig neues Verzeichnis angelegt, nehmen wir »/tftpboot«. In diesem wird nun das Unterverzeichnis »root« erstellt, in das wir mit *tar -x -f /pfad/zu/diskimage.tgz -C /tftpboot/root -z* das Diskimage entpacken. Nun noch den Kernel nach »/tftpboot« kopiert und mit *gunzip netbsd-INSTALL.ecoff.gz* entpackt und

am besten gleich noch mit *chmod a+r netbsd-INSTALL.ecoff* Leserechte für alle User gesetzt. Beides sollte als User *root* geschehen, damit die einzelnen Benutzerrechte im Diskimage unangetastet bleiben.

Als nächstes müssen die einzelnen Server-Prozesse konfiguriert werden. Eine netzbootende DECStation versucht zuerst, über das BOOTP-Protokoll an eine IP-Adresse und den Namen des zu verwendenden TFTP-Servers samt zu ladendem Dateinamen zu gelangen. Der TFTP-Server muß diese Datei natürlich vorrätig haben und auch zur Verfügung stellen. Als letzter Schritt muß nun im Netzwerk ein NFS-Server sein, der das Installations-Diskimage im Netz freigibt. All diese Dienste laufen für gewöhnlich auf ein- und demselben Rechner, könnten jedoch auch auf verschiedenen Maschinen laufen. Um den Rahmen dieses Kapitels nicht zu sprengen, möchte ich nur die Konfiguration dieser Serverdienste unter den BSD-Unixen und Linux ansprechen. Unter anderen Systemen sind die Eckdaten ähnlich, genaue Anweisungen entnimmt man am besten dem »Diskless NetBSD Howto« unter [7].

Zuerst zum BOOTP-Server. NetBSD, OpenBSD und FreeBSD liefern einen *bootp*-Server namens *bootpd* mit. Die Konfigurationsdatei des BOOTP-Servers heißt »/etc/bootptab« und hat folgendes Format:

```
pmax.testnetzwerk.de  :\
                      :ht=ether:\
                      :ha=CCCCCCCCCCCC:\
                      :sm=255.255.255.0:\
                      :lg=192.168.1.5:\
                      :ip=192.168.1.10:\
                      :hn:\
                      :bf=netbsd-INSTALL.ecoff:\
                      :bs=auto:\
                      :rp=/tftpboot/root/:
```

Statt *pmax.testnetzwerk.de* sollte hier natürlich der Name der DECStation stehen, dieser kann beliebig gewählt werden. Die Reihe von »C« im Feld *ha* steht für die Hardware-Adresse der DECStation, die mit dem *cnfg*-Bootmonitorbefehl eingesehen werden kann – sie wird in diesem Fall ohne die Trennzeichen »:« oder »-« geschrieben. Die Felder *ip* und *sm* stehen für die IP-Adresse und zugehörige Subnetzmaske, die der Workstation mitgeteilt werden soll – diese sollte sich natürlich im gleichen Netz wie der Bootserver befinden. Die Adresse hinter *lg* steht für das Local Gateway, also das Gateway des Netzes. Ist kein Router in andere Netze oder ins Internet vorhanden, kann dieses Feld auch einfach leer gelassen werden. Wird der Standard-Netzkernel von NetBSD 1.6 nicht verwendet, muß der Dateiname des Kernels im Feld *bf* eventuell angepaßt werden. Nun muß lediglich in der Datei »/etc/inetd.conf« dem *inetd* mitgeteilt werden, daß er den *bootpd* auch starten soll. Dies gelingt unter NetBSD und OpenBSD durch das Löschen der Raute »#« vor der folgenden Zeile:

```
#bootps dgram udp wait root /usr/sbin/bootpd bootpd
```

Durch Eingabe von *ps ax|grep inetd* wird die PID des *inetd* angezeigt. Mit *kill -HUP PID* wird der *inetd* neu gestartet. Zeigt der *ps*-Befehl keinen *inetd* an, kann er einfach mit *inetd* gestartet werden. All dies muß natürlich als *root* erfolgen. Treten bei einem dieser Schritte Probleme auf, hilft meist das »Diskless NetBSD Howto« weiter [7].

Bei der Konfiguration des TFTP-Servers genügt es, dem System mitzuteilen, daß der *tftpd* gestartet werden soll, sobald entsprechende Anfragen eintrudeln. Dazu sollte vor folgender Zeile der »/etc/inetd.conf« das Kommentarzeichen »#« entfernt werden:

```
#tftp dgram  udp wait root /usr/libexec/tftpd tftpd -s /tftpboot
```

Der *inetd* muß nun genau wie beim *bootpd* neu gestartet werden.

Nun sollte der *tftpd* getestet werden. Folgende Sequenz lädt den Kernel nach »/tmp« und wird auf dem gleichen Rechner ausgeführt, auf dem auch der *tftpd* läuft.

```
# cd /tmp
# tftp localhost
tftp> get netbsd-INSTALL.ecoff
Received 4417376 bytes in 1.3 seconds
tftp> quit
```

Liefert *tftp* statt der Received-Meldung einen Fehler zurück, sollte die Konfiguration des *tftpd* nochmal genau überprüft werden. Bei einem »File not Found«-Fehler ist es möglich, daß der verwendete *tftp*-Server zur Angabe des Dateinamens auch den kompletten Pfad erwartet. In der »bootptab« (beziehungsweise in der »dhcpd.conf«) sollte also *netbsd-INSTALL.ecoff* durch */tftpboot/netbsd-INSTALL.ecoff* ersetzt werden.

Nun fehlt nur noch die NFS-Konfiguration. Es genügt folgender Eintrag in der »/etc/exports«:

```
/tftpboot/root -maproot=root:wheel 192.168.1.8
```

Wie immer ist hier 192.168.1.8 durch die IP-Adresse der DECStation zu ersetzen. Nun müssen die NFS-Serverprozesse neu gestartet werden. Gibt ein *ps -ax|grep mountd* die PID eines Prozesses aus, kann dieser mit *kill -HUP PID* neu gestartet werden. Dasselbe muß mit *ps -ax|grep nfsd* für den zweiten Prozeß gemacht werden. Sind die Prozesse nicht aktiv, können sie von Hand durch Eingabe von */usr/sbin/mountd* gestartet werden.

Bevor nun mit der eigentlichen Installation begonnen werden kann, muß noch eine kleine Vorsorge gegen einen Bug in vielen Bootmonitor-Versionen der DECStation getroffen werden. Oft antworten sie nicht auf ARP-Anfragen, sobald die IP-Adresse per BOOTP zugewiesen wurde. So weiß der TFTP-Server nicht, an welchen Rechner er die Daten zu senden hat. Dies kann einfach behoben werden, indem der ARP-Eintrag fest verdrahtet wird. Unter den meisten Betriebssystemen genügt ein *arp -s 192.168.1.8 CC:CC:CC:CC:CC:CC*, dies trifft auch auf die BSD-Systeme zu.

Nun wird über den Bootmonitor-Befehl *boot 3/tftp* das System gestartet. Die Slotnummer hängt dabei natürlich von der jeweiligen Konfiguration ab. Es empfiehlt sich, ständig die Logfiles des Bootservers im Auge zu behalten. Einige Bootmonitor-Versionen haben Probleme damit, große Dateien zu starten. In diesem Fall ist es empfehlenswert, zuerst NetBSD 1.5.2 zu installieren und dann eventuell später ein lokales Update durchzuführen.

2.5.2 Installation über eine Festplatte oder ein CD-ROM

Wenn eine übrige SCSI-Platte und ein zweiter Rechner mit SCSI-Bus und einem installierten BSD-Unix vorhanden sind, kann das Miniroot auch direkt auf die Platte gespielt werden. Dieses Vorgehen ist jedoch unter Umständen schwierig und setzt mehr Kenntnisse voraus als der Weg über Netbooting. Es muß das Diskimage eines gesamten kleinen Installationsdateisystems heruntergeladen werden. Es liegt im Verzeichnis »installation/diskimage/diskimage.gz« auf dem FTP-Server, der ganze Pfad ist in [12] notiert. Dieses Image wird mit *gunzip* entpackt und dann mit dem Befehl *dd* auf die Platte aufgespielt. Vorher muß die Platte noch ein funktionierendes Disklabel erhalten. Auf der PC-Architektur sollte das Label mit *disklabel -W /dev/rsdXd*, auf allen anderen Plattformen mit *disklabel -W /dev/rsdXc* geschrieben werden. *X* ist die Nummer der zu verwendenden SCSI-Platte. Höchste Vorsicht ist geboten, daß nicht aus Versehen ein wichtiges Dateisystem zerstört oder gelöscht wird! Nun kann das Image mit *gunzip -c diskimage.gz | dd of=/dev/rsdXc bs=10240* (beziehungsweise *rsdXd*) aufgespielt werden.

Die Installation über ein CD-ROM ist an sich der komfortabelste Weg. Allerdings haben DECStations die gleiche Einschränkung wie die meisten Workstations: Verwendete SCSI-CD-ROMs müssen den Betrieb mit 512 Byte großen Sektoren unterstützen. Die meisten aktuellen CD-ROMs können dies nicht. Leider hilft hier nur ausprobieren. Das entpackte Image wird hierfür »roh« direkt auf den Rohling gebrannt, etwa mit *cdrecord*. Danach ist der Bootvorgang derselbe wie für Festplatten.

Nun muß die Platte wieder in die DECStation gebaut werden. Mit *cnfg* läßt sich leicht der Name des entsprechenden Devices für den Bootmonitor heraussuchen. Danach kann das Installationssystem mit *boot 3/rzX/netbsd* gestartet werden. Sollte ein SCSI-Controller in einem anderen Slot verwendet werden, bitte die Slotnummer entsprechend der Angabe von *cnfg* anpassen. *rzX* bezieht sich auf den Bootmonitor-Devicenamen der Festplatte beziehungsweise des CD-ROMs.

2.5.3 So geht es weiter

Die Bootmeldungen von NetBSD-pmax 1.6 auf einer DECStation 5000/133 sind in Abbildung 2.49 dargestellt.

Der wesentliche Unterschied zu älteren Versionen ist, daß NetBSD vor 1.5.3 nicht das plattformübergreifende MI-SCSI verwendete, sondern das »alte« SCSI-System. Die SCSI-Devices heißen dementsprechend nicht *sd*, sondern *rz*.

Nach dem Start des Kernels werden Sie von der menügesteuerten sysinst-Installation erwartet. Das mächtige Hilfsprogramm ist auf allen NetBSD-Plattformen gleich! Die Installation kann wie gewohnt betrieben werden.

```
>>boot 3/tftp
4412864+0+218748
segment  0 start 00000000 size 04000000
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All rights reserved.

NetBSD 1.6 (RAMDISK) #0: Mon Sep  9 03:01:25 UTC 2002
    autobuild@tgm.daemon.org:/autobuild/pmax/OBJ/autobuild/src/sys/arch/pmax/
compile/RAMDISK
DECstation 5000/133 (3MIN)
total memory = 65536 KB
avail memory = 55700 KB
using 844 buffers containing 3376 KB of memory
mainbus0 (root)
cpu0 at mainbus0: MIPS R3000A CPU (0x230) Rev. 3.0 with MIPS R3010 FPC Rev. 4.0
cpu0: 64KB/4B direct-mapped Instruction cache, 64 TLB entries
cpu0: 128KB/4B direct-mapped write-through Data cache
tc0 at mainbus0: 12.5 MHz clock
ioasic0 at tc0 s lot 3 offset 0x
le0 at ioasic0 offset 0xc0000: address 08:00:2b:34:ed:e4
le0: 32 receive buffers, 8 transmit buffers
scc0 at ioasic0 offset 0x100000
scc1 at ioasic0 offset 0x180000: console
mcclock0 at ioasic0 offset 0x200000: mc146818 or compatible
asc0 at ioasic0 offset 0x300000: NCR53C94, 25MHz, SCSI ID 7
scsibus0 at asc0: 8 targets, 8 luns per target
scsibus0: waiting 2 seconds for devices to settle...
sd0 at scsibus0 target 0 lun 0: <CONNER, CFP4207S  4.28GB, 1524> SCSI2 0/direct
fixed
sd0: 4096 MB, 3999 cyl, 20 head, 104 sec, 512 bytes/sect x 8388608 sectors
sd0: sync (200.0ns offset 15), 8-bit (5.000MB/s) transfers, tagged queueing
sd1 at scsibus0 target 3 lun 0: <DEC, RZ25     (C) DEC, 0900> SCSI2 0/direct fixed
sd1: 406 MB, 1476 cyl, 9 head, 62 sec, 512 bytes/sect x 832527 sectors
sd1: sync (200.0ns offset 15), 8-bit (5.000MB/s) transfers
md0: internal 3072 KB image area
boot device: le0
root on md0a dumps on md0b
WARNING: clock gained 96 days -- CHECK AND RESET THE DATE!
root file system type: ffs
Terminal type? [rcons] vt100

 Welcome to sysinst, the NetBSD-1.6 system installation tool.
```

Abb. 2.49: Bootmeldungen von NetBSD-pmax 1.6 auf einer DECStation 5000/133

Alles in allem ist eine DECStation ein gutes, solides Unix-System. Die CPU-Leistung mag heutzutage neben den Gigahertz-Boliden verblassen, aber die I/O-Leistung ist für eine Maschine dieses Alters beeindruckend und genügt auch vielen aktuellen Anwendungen. So kann die DECStation 5000/133 beispielsweise in achtfacher Geschwindigkeit eine CD brennen oder über die eingebaute Netzwerkkarte zirka 900 Kilobyte pro Sekunde über das Netzwerk übertragen. Auch sind diese Maschinen sehr stabil, Abstürze kommen bei gesunder Hardware im Jahrestakt vor.

boot 3/tftp	Sucht einen BOOTP- und TFTP-Server im Netzwerk über die Netzwerkkarte in Slot 3 (was bei DECStations 5000 die Onboard-Netzwerkkarte ist).
boot 0/rz1/netbsd	Startet auf der SCSI-Platte (oder Wechselmedium) mit der SCSI-ID 1 die Datei »netbsd«. Das Gerät hängt am SCSI-Bus des Controllers, der in Turbochannel-Slot 0 steckt.
boot	Verwendet die Informationen in der boot-Umgebungsvariable zum Starten.

Tabelle 2.5: Beispiele für Bootbefehle

Literatur zur pmax-Installation

[1] Bemerkungen zu den einzelnen PMAX-Modellen:
http://www.netbsd.org/Ports/pmax/models.html

[2] Informationen zu Board- und ROM-Versionen:
http://www.netbsd.org/Ports/pmax/board-list.html

[3] DECStation 5000/xx Handbuch:
http://decstation.unix-ag.org/docs/dec_docs/maxine.ps.gz

[4] NetBSD/pmax FAQ: *http://www.netbsd.org/Ports/pmax/faq.html*

[5] NetBSD FTP-Server pmax Diskimage Netzwerk: *ftp://ftp.de.netbsd.org/pub/NetBSD/NetBSD-1.6/pmax/installation/netboot/diskimage.tgz*

[6] NetBSD FTP-Server pmax Netzboot-Kernel: *ftp://ftp.de.netbsd.org/pub/NetBSD/NetBSD-1.6/pmax/binary/kernel/netbsd-INSTALL.ecoff.gz*

[7] Diskless NetBSD HOWTO:
http://www.netbsd.org/Documentation/network/netboot/

[8] Quelltext von bootpd:
ftp://ftp.uni-duisburg.de/Unix/networking/bootp-DD2.4.3.tar.gz

[9] GNU Inetutils: *http://www.gnu.org/software/inetutils/inetutils.html*

[10] Netzwerkinstallation mit Windows:
http://www.afn.org/~raub/netbsd/bootpw95.html

[11] utftpd Homepage: *http://www.ohse.de/uwe/software/utftpd.html*

[12] NetBSD FTP-Server pmax Diskimage SCSI: *ftp://ftp.de.netbsd.org/pub/NetBSD/NetBSD-1.6/pmax/installation/diskimage/diskimage.gz*

3 NetBSD konfigurieren

3.1 Grundkonfiguration des Systems

von Thorsten Lindloff

3.1.1 Der Aufbau des NetBSD-Verzeichnisbaums

Im Gegensatz zu Microsoft-Systemen kennt NetBSD wie alle Unix-Derivate keine Laufwerksbuchstaben im klassischen Sinn. Es gibt nur einen Verzeichnisbaum, in den ein Laufwerk eingehängt werden kann. Um welche physikalische Gerätebauart es sich bei einem Gerät handelt, das irgendwo an den Verzeichnisbaum gehängt wird, ist NetBSD, wie auch allen anderen Unix-Systemen, egal. Relevant ist nur, ob die Dateisysteme nur lesbar oder les- und beschreibbar sind.

Alle freien Unix-Derivate besitzen neben dem System der Konfiguration mit Textdateien[1] eine weitere Gemeinsamkeit: die Verzeichnisstruktur ist prinzipiell austauschbar. Die Struktur des Baums unter NetBSD zeigt Tabelle 3.1.

Eine CD mounten

Müssen Sie von CD-ROM nachinstallieren, geschieht das mit dem Kommando *mount*, das unter NetBSD mit ein paar Default-Parametern geschmückt ist, um dem Benutzer das Eingeben des Kommandos zu erleichtern. Mit diesem Befehl wird ein ATAPI-CD-ROM-Laufwerk in das Dateisystem eingehängt. Das Verzeichnis, unter dem die Daten des Laufwerks auftauchen, ist »/mnt«:

```
#mount_cd9660 /dev/cd0a /mnt
```

3.1.2 vi, das wichtigste Programm

Die wichtigste Frage zuerst: Warum wird dieser »Editorgott« namens vi, dem schon Generationen von Informatikern geopfert wurden, überall verwendet, wo Unix oder ähnliches läuft? Die Antwort darauf ist ganz einfach: vi ist recht kompakt und weil er kompakt ist, ist er schnell. Ein weiterer Vorteil ist, daß man diesen Editor oder einen seiner Clones (bei NetBSD ist das »nvi«) auf jedem Unix-System oder einem seiner Abkömmlinge vorfindet. Er wird bei der Installation auf jedes System gespielt. Es gibt Mitmenschen, die in diesem Editor tatsächlich eine Art »Gott« sehen und ausschließlich mit ihm arbeiten, und es gibt die anderen, die ihn überhaupt nicht mögen. Beide Seiten haben gute Gründe für ihre jeweilige Meinung. Wenn Sie vi nicht mögen, kann »joe« eine Alternative dazu sein. Er läßt sich wie WordStar bedienen, was viele Anwender der alten Schule als angenehm empfinden.

---

[1] Man nennt dieses System »rc-System«, weil die zentrale Konfiguration der Ressourcen über eine Datei »rc« durchgeführt wird.

Auch der wesentliche Nachteil dieses Editors ist allgemein bekannt, nämlich die sehr merkwürdige Bedienung, die insbesondere für Anwender, die bisher nur mit Word oder ähnlichem Bekanntschaft gemacht haben, gewöhnungsbedürftig ist.

Eintrag	Inhalt
/bin	Alle Dateien, die von allen Benutzern ausgeführt werden dürfen.
/dev	Die Gerätedaten.
/etc	Die Konfigurationsdateien des Systems.
/home	Die Daten der Benutzer außer root.
/kern	Entspricht im wesentlichen in der Funktionsweise dem »/proc«-Dateisystem unter Linux.
/mnt	Das Verzeichnis, in das üblicherweise »fremde« Dateisysteme eingehängt werden, die der Kernel standardmäßig nicht mountet. Ein CD-ROM-Laufwerk oder eines aus dem Netzwerk sind solche Laufwerke.
netbsd	Der Kernel.
/root	Der Name ist Zeichen: Heimatverzeichnis des Administrators
/sbin	Programme, die nur der Systemadministrator *root* ausführen darf.
/stand	Ist nach der Installation leer.
/tmp	Temporäre Daten.
/usr	Dieses Verzeichnis hat in NetBSD noch eine weitere Funktion: Hier befinden sich neben zusätzlichen Programmen die Daten aus dem Package-System von NetBSD.
/var	An diesem Ort werden Logdaten, Mails, Datenbankdaten und anderes abgelegt oder zwischengelagert. Als Eselsbrücke: der Inhalt ist variabel.

Tabelle 3.1: Die Einträge im Hauptverzeichnis bei NetBSD

Starten Sie diesen Editor einmal probehalber und erstellen Sie eine Textdatei mit ihm:

```
#vi meine_Datei
```

Anschließend drücken Sie die Taste [Einfg]. Jetzt wartet vi darauf, Ihre Gedanken entgegenzunehmen. Solange Sie sich nicht verschreiben, ist die Sache recht einfach – Wenn Sie einen Absatz beendet haben, drücken Sie die Eingabetaste und schreiben in der neuen Zeile normal weiter. Wenn Sie sich verschrieben haben, navigieren Sie an den Anfang des Worts und löschen Sie es mit der Taste [Entf]. Anschließend drücken Sie wieder [Einfg] und schreiben die korrigierte Version des Worts in den Text.

Will man eine ganze Zeile löschen, navigiert man mit den Cursortasten in die betreffende Zeile und drückt [Esc] [d] [d].

Wenn Sie dort wieder etwas hineinschreiben wollen, gehen Sie an das Ende der vorherigen Zeile, drücken auf [Einfg] und die Eingabetaste.

Sie wollen den geschriebenen Text speichern: [Esc] [:] [w].

Ist die Datei schreibgeschützt, erzwingt das [!] hinter dem [w], daß sie trotzdem geschrieben wird.

Speichern und beenden: [Esc] [:] [w] [q].

Abbrechen bewirkt [Esc] [:] [q] [!]. Das Ausrufezeichen sagt vi, daß das Programm ohne Speichern der Änderungen beendet werden soll.

Das sind die wesentlichen Tastenkombinationen des vi. Als Einstieg reichen die besprochenen Befehle aus, für mehr konsultieren Sie die Manpages zu vi (*man vi*).

3.1.3 Die Tastatur an lokale Gegebenheiten anpassen

Wenn das System hochgefahren ist, erscheint eine Meldung »login:«.

dk	Dänemark
fr	Frankreich
it	Italien
fr	Frankreich
jp	Japan
sv	Schweden
uk	Das Vereinigte Königreich
us	USA. Das ist die Standardbelegung.

Tabelle 3.2: Die wichtigesten Tastaturbelegungskürzel

Geben Sie hier »root« ein, bestätigen Sie das mit der Eingabetaste und setzen Sie anschließend Ihr Passwort ab. Sie werden aufgefordert, einen Non-root-Account anzulegen. Das ist ein normaler Benutzer für die tägliche Arbeit. Er besitzt weniger Rechte als der Systemadministrator *root* und reicht für die normale Arbeit aus. Generell gilt, daß man bei Multi-User-Systemen – auch wenn man nur zuhause damit arbeitet – nie als Administrator arbeiten sollte, außer natürlich für Administrationsaufgaben. Die Gefahr, unbedacht das System zu zerstören, ist zu groß. Lassen Sie es an diesem Punkt bei der Meldung bewenden, darum kümmern Sie sich später. Zuerst einmal brauchen Sie eine zu Ihrem Land passende Tastaturbelegung. Wenn Sie eine »deutsche« Tastatur besitzen, ist

```
#wsconsctl -k -w encoding=de
```

die richtige Wahl. Weitere Länderkennungen, die sehr verbreitet sind, sind in Tabelle 3.2 aufgeführt.

Einige Länderkennung wie beispielsweise die deutsche erlauben bei der Tastaturdefinition noch eine Zusatzangabe *nodead*, der Befehl sieht dann so aus:

```
#wsconsctl -k -w encoding=de.nodead
```

Damit werden die »Tottasten« abgeschaltet. Tottasten sind beispielsweise die Akzenttasten. Sie ergeben in Kombination mit einer zweiten Eingabe ein zusammengesetztes Zeichen. Die Eingabe von [^][e] ergibt also ê. Nicht allein die Akzenttasten sind solche Tottasten, auch die Tilde (~). Da viele Editoren unter unixartigen Betriebssystemen als Backup-Kennung ein Tilde-Zeichen verwenden, ist es recht angenehm, wenn man beim Löschen dieser Backup-Dateien nicht erst *rm* *~ mit einem anschließenden Leerzeichen angeben muß, um die Tilde zu erhalten. Es ist nämlich recht peinlich, wenn man sich darauf verläßt, daß die Tilde schon dasteht und [Enter] betätigt. Aus dem Löschen aller Backup-Dateien im aktuellen Verzeichnis wird dann nämlich schnell ein Löschen *aller* Dateien im Verzeichnis. Weitere Daten zur Tastaturbelegung und weitere Länderkennungen finden Sie in »/sys/dev/wscons/wsksymdef.h« ab Zeile 458.

Dann ist ein Wechsel in das Verzeichnis »/etc« angesagt. Das geht so:

```
#cd /etc
```

Jetzt bearbeiten Sie die Datei »rc.local«[1] mit dem Editor »vi«.

```
#vi rc.local
```

Nochmals:

Drücken Sie die Taste [Einfg]. Gehen Sie an das Ende der letzten Zeile und drücken Sie die Eingabetaste. Geben Sie das oben genannte Kommando noch einmal ein und beenden Sie vi mit [Esc] [:] [w] [q].

Wenn irgend etwas nicht geklappt haben sollte und Sie die geänderte Datei nicht speichern wollen, geht auch das: [Esc] [:] [q] [!] beendet vi, ohne daß die Änderungen in die Datei geschrieben werden.

Mit dem *echo*-Befehl tragen Sie dieses Kommando in »rc.local« ein, wenn Sie sich mit dem Editor vi nicht einig werden:

---

[1] Das Ganze geht genauso mit der allgemeinen Steuerdatei »rc.conf«. Die »rc.local« zu verwenden, ist aber eigentlich der ordentliche Weg.

```
#echo "wsconsctl -k -w encoding=de" >>/etc/rc.local.
```

Zu Ihrer Sicherheit: Das bewirkt das Gleiche wie der manuelle Eintrag mit vi, ist allerdings risikobehaftet. Es müssen immer zwei »>>« eingegeben werden, sonst wird alles andere in dieser Datei gelöscht, was zu Folge hätte, daß man unter Umständen Daten verliert. Das ist in der »rc.local« nach einer Neuinstallation noch nicht von allzu großer Bedeutung, aber wenn das System wächst, kann sich das schnell ändern. Außerdem können Sie sich so leichter an eine schnellere Systemkonfiguration nach diesem Muster gewöhnen.

Lösung Nummer Zwei ist der Neubau eines Kernels, in den die entsprechende Tastaturbelegung mit einkompiliert wird. Das soll an dieser Stelle nicht besprochen werden, ich verweise auf das entsprechende Kapitel im Buch.

3.1.4 Das Passwort

Nun sollte man das root-Passwort ändern beziehungsweise überhaupt festlegen, falls das bei der Installation noch nicht getan wurde. Geben Sie den Befehl *passwd* ein. Dann erscheint folgendes auf dem Monitor:

```
Changing local Password for root.
New Password:
Retype new password:
```

Sie haben sich das Passwort gemerkt? Falls nicht, haben Sie immer noch die Chance, das Passwort zu ändern, solange Sie noch eingeloggt sind.

3.1.5 Wechseln der Shell

Das nächste, das Sie unter Umständen vornehmen wollen, ist ein Wechsel der Shell. NetBSD wird immer mit der *csh*, einer C-Shell, installiert. Dies ist die Standardshell, die für den User *root* vorinstalliert ist. Vielleicht besitzen Sie ja noch ein anderes System, auf dem vielleicht eine Bash oder eine sh läuft, an die Sie bereits gewöhnt sind.

Das entsprechende Kommando für den Wechsel der Shell ist *chsh* (für »change shell«). Grundsätzlich hat man in NetBSD die Auswahl zwischen drei vorinstallierten Shells, der csh, ksh und sh.

Geben Sie jetzt folgendes ein:

```
chsh
```

In der Datei, die nun geöffnet wird, stehen Ihre Benutzerdaten und welche Shell verwendet wird. Die dort eingetragene Shell wird geändert, indem hinter dem Eintrag »Shell: /bin/tcsh« ein anderer Dateiname angegeben wird. Generell muß die entsprechende Shell vorher in der Datei »/etc/shells« eingetragen sein. Nur diese Dateien werden akzeptiert. Falls Sie einen beliebigen Namen eingeben, warnt Sie das Konfigurati-

onsprogramm[1]. Es reicht auch nicht, nur den Namen einzugeben, es ist immer der komplette Pfad ebenfalls einzutragen, siehe Abbildung 3.1.

Der Editor, der auf eine Eingabe wartet, ist voreingestellt vi. Ist Ihnen vi zu krude, läßt sich der Systemeditor durch Umdefinition der Umgebungsvariable *EDITOR* auf jeden beliebigen anderen Editor setzen. Das machen Sie, indem Sie bei Verwendung der *csh* oder *tcsh* diesen Befehl eingeben:

```
setenv EDITOR Pfad_und_Name_zum_Editor
```

Mit der *bash*, *ksh* oder *sh* lautet der Befehl:

```
export EDITOR=Pfad_und_Name_zum_Editor
```

```
#Changing user database information for root.
Login: root
Password: OJfSY3UmWyuJk
Uid [#]: 0
Gid [# or name]: 0
Change [month day year]:
Expire [month day year]:
Class:
Home directory: /root
Shell: /usr/pkg/bin/bash
Full Name: Charlie &
Location:
Office Phone:
Home Phone:
~
~
~
~
~
~
~
~
~
/etc/pw.00753a: unmodified: line 1
```

Abb. 3.1: Der Aufruf von chsh. Als Shell ist hier die bash installiert

Die C-Shell (csh, Berkeley C-Shell) ist als solches noch nicht einmal schlecht. Sie hat nur einen kleinen Makel: Der Umsteiger von Linux kommt damit schlecht zurecht, weil sie nicht kompatibel zur dort verwendeten Bash ist. Ihr größter Vorteil ist eigentlich, daß damit in einer der Programmiersprache C ähnlichen Syntax programmiert werden kann, was den Gewohnheitstieren unter den Unix-Anwendern häufig die Arbeit erleichtert.

---

[1] Das Package-System von NetBSD trägt bei der Installation von Shells den Namen automatisch ein. Sie brauchen Sie nicht darum zu kümmern. Nur wenn Sie eine Shell manuell installieren, müssen Sie darauf achten. Wenn Sie das Einloggen eines Benutzers damit verhindern wollen, daß als Shell das Programm *nologin* eingetragen ist, müssen Sie »/sbin/nologin« der Shelldefinitionsdatei zufügen.

Wenn Sie Ihre neue Shell csh, oder genauer: ihr Verhalten anders konfigurieren wollen, tun Sie das in der Datei ».cshrc« im Heimatverzeichnis. Für die anderen Shells gilt das Gleiche.

Darüber hinaus sollten Sie noch die Systemzeit kontrollieren. Das geht folgendermaßen:

Prüfen Sie, ob die örtliche Zeitzone richtig gesetzt ist. Rufen Sie dazu den Befehl *date* auf. Es sollte dann eine Ausschrift zu sehen sein, bei der als Zeitzone CET (Winterzeit) oder CEST zu lesen ist. Ist das der Fall, kann man davon ausgehen, daß die mitteleuropäische Zeit eingestellt ist. Sie können natürlich auch prüfen, ob die Datei »etc/localtime« auf die richtige Zeitzonendatei zeigt. Eine Ausschrift wie folgt sollte zu sehen sein:

```
lrwxr-xr-x 1 root wheel 33 Nov 15 17:56 localtime ->
/usr/share/zoneinfo/Europe/Berlin
```

was anzeigt, daß die richtige Zeitzonendatei verwendet wird. Haben Sie einen Ausgabe wie folgt:

```
-r--r--r-- 1 root  wheel  837 Nov 15 17:59 localtime
```

bedeutet das, daß anstelle eines Links die Datei kopiert wurde. Hier ist es nicht aus der Datei direkt zu erfahren, auf welche Zeitzone sie zeigt. Löschen Sie sie und legen Sie einen neuen Link.

Auch wenn »Europe/Paris« oder eine andere Stadt aus der mitteleuropäischen Zeitzone steht, wurde der Link bei der Installation korrekt gesetzt. Suchen Sie sich einfach Ihre Lieblingsstadt in Ihrer Zeitzone aus. Sie können auch einmal die anderen Zeitzonendateien ansehen, sie befinden sich in »/usr/share/zoneinfo« und sind dort in Verzeichnisse für die jeweiligen Kontinente eingeteilt.

Alle Unix-Systeme gehen davon aus, daß die Rechneruhr nicht in der lokalen Zeit läuft, sondern auf UTC (Universal Time Coordinates, Greenwich-Zeit) eingestellt ist. Ziemlich alle Rechner halten sich auch daran, natürlich nicht der Intel-PC. Erst mit den neuesten Windows-Versionen ab 2000 ist es überhaupt möglich, dem Betriebssystem aus Redmond klar zu machen, daß eine Uhr von UTC auf lokale Zeit umgerechnet werden soll. Deshalb machen praktisch alle Hardware-Uhren auf intel-basierten Systemen mit Unix Ärger. Wenn Sie außer NetBSD kein anderes Betriebssystem oder höchstens ein anderes Unix betreiben, sollten Sie die Uhr unix-konform im BIOS umschalten. Wenn Sie zu den Unglücklichen gehören, die nicht ganz auf Windows verzichten können, müssen Sie Ihren NetBSD-Kernel patchen, um ihm die Zeitdifferenz zwischen der kleinen Stadt in England und Ihrem Wohnort bekannt zu geben.[1] Dann müssen Sie die in der Hardwareuhr eingestellte Zeit dem Kernel bekannt machen. Das geht so:

---

[1] Sie können die Zeitdifferenz zu Greenwich auch, wenn sie einen eigenen Kernel bauen, diesem in der Konfigurationsdatei von vornherein mitgeben. Sie sparen sich dann den Patch.

Fragen Sie den Kernel-Offset ab. Das ist die Zeitspanne in Minuten, die die Zeit am Nullmeridian in Greenwich gegenüber der Ortszeit »nachgeht«

```
sysctl kern.rtc_offset
```

gibt die folgende Meldung aus:

```
kern.rtc_offset = 0
```

Jetzt ist es an der Zeit, ein wenig am Kernel zu arbeiten und ihn zu patchen. Geben Sie als *root* folgendes ein:

```
# gdb --write /netbsd
GNU gdb 4.17
Copyright 1998 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public
License, and you are welcome to change it and/or distribute
copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.
Type "show warranty" for details.
This GDB was configured as "i386--netbsd"...
(no debugging symbols found)...
(gdb) set rtc_offset=-60
(gdb) quit
#
```

Sie haben hiermit den GNU Debugger gestartet und die Datei »/netbsd« mit Schreibberechtigung geöffnet, also den aktuellen Kernel. Haben Sie dem Kernel einen anderen Namen gegeben, sollten Sie diesen nehmen. Dann haben Sie im *gdb* mit der Zeile *set rtc_offset = -60* den Offset in Minuten *westlich* von Greenwich eingetragen. Da sich Mitteleuropa bekanntermaßen östlich von Greenwich befindet, wird vor dem Zeitoffset das Minuszeichen angegeben.

Wenn Sie den Debugger wie im Listing gezeigt mit *quit* verlassen, wird die geänderte Kerneldatei automatisch gespeichert. Diese Änderung ist permanent, muß aber für jeden neu gebauten Kernel, bei dem der Offset nicht in der Konfigurationsdatei eingetragen ist, neu vorgenommen werden. Die Änderung der Uhrzeit zeigt erst nach dem nächsten Neustart des Systems Wirkung.

3.1.6 Weitere Benutzer anlegen

Aus Sicherheitsgründen ist es ratsam, neben dem Systemverwalter *root* noch mindestens ein weiteres Benutzerkonto anzulegen. Das empfiehlt sich auch, wenn Sie der einzige Benutzer des Systems sind. Sie können dann mit einem Kommando *root* werden.

Für dieses Vorgehen sprechen Datenschutzgründe und die Ausfallsicherheit des Systems, denn auf einem Single-User-System wie Windows gibt es nur einen Benutzer, der alle Arbeiten erledigt. Er ist Anwender, Softwareentwickler und Administrator in einer Person. Solch eine Konstellation mag bei einem System, das im heimischen Schlafzimmer steht und nur von einer Person genutzt wird, noch gutgehen. Aber die Ruhe ist meistens schon vorbei, wenn sich Familienzuwachs einstellt. Dann kann das schon problematisch werden, wenn der Filius die Kontodaten von Papa kennt und dort vielleicht auch noch eine Liste mit den wirklich heiklen Transaktionsnummern der Bank herumliegt oder vielleicht eine Kreditkartennummer oder ähnliches.

Neue Benutzerkonten anlegen

Wer einen Computer hat, hat meist auch etwas zu verbergen oder geheimzuhalten. Das ist natürlich auf einem Single-User-System nicht so ganz einfach wie bei einem Unix-Verwandten.

Lange Rede, kurzer Sinn, man braucht aus den genannten Gründen neben *root* noch mindestens einen Benutzer, unter dessen Konto man die alltägliche Arbeit ausführt.

Das Kommando zum Anlegen eines neuen Benutzers ist:

```
#useradd -m musterfred
```

Dann geben Sie folgendes ein:

```
#passwd musterfred
```

Antwort:

```
#enter new password: ********
#re-enter new password: ********
```

Diese Prozedur löst folgende Vorgänge aus:

- Ein neuer Benutzer wird in »/etc/passwd« und »/etc/group« gelistet.
- Der neue Benutzer wird einer Gruppe hinzugefügt. Standardmäßig ist das die Gruppe *users*« mit der GID 500 (GID bedeutet »Gruppen-Identifizierungsnummer«).
- Für den neuen Benutzer wird ein Homeverzeichnis angelegt. Darauf kann unter Umständen auch verzichtet werden, indem man das *-m* hinter dem Befehl *useradd* wegläßt.

In dieses Verzeichnis werden die für seine Arbeit nötigen Konfigurationsdateien kopiert. Das sind im wesentlichen:

- .shrc oder .cshrc, je nach Shell
- .mailrc

- .login
- .mailrc
- .profile
- .rhosts

Wenn X eingerichtet ist, kommen abhängig vom Window Manager noch einige Dateien dazu. Das sind:

- .Xinitrc
- .twmrc

Diese Dateien sollte außer dem Eigentümer (also dem neuen Benutzer) und *root* niemand lesen können. Auch das ist voreingestellt so konfiguriert.

Gruppen

Wenn Sie Ihren zweiten Benutzer angelegt haben, öffnen Sie mit *vi* die Datei »/etc/group«, in der ein paar Benutzergruppen eingetragen sind. Die für uns interessante ist *wheel*. Wer dort eingetragen ist, hat das Recht, *root*, also Systemverwalter, zu werden. Tragen Sie am Ende der Zeile einfach den Benutzernamen ein. Das Ergebnis sollte so aussehen:

```
wheel:*:0:root,NeuerUsername
```

Wenn das getan ist, können Sie mit Ihrem Standard-Benutzer *root* werden. Das geht über den Befehl *su*.

```
$su
Password:
Sorry
$su
Password:
#
```

Geben Sie das *root*-Passwort ein, und schon sind Sie Systemverwalter. Das *su*-Kommando funktioniert auch andersherum, wenn Sie *root* sind oder Ihr Standardbenutzer die Berechtigung hat, mit diesem Kommando den Namen zu wechseln. *su* ist übrigens die Abkürzung für »switch user«. Sinn macht diese Art der Verwaltung auch, wenn Sie ein Projekt planen, für das die NetBSD-Maschine als Server dienen soll. Dafür können Sie auch eine neue Gruppe anlegen:

```
#groupadd projekt
```

Wenn Sie nun einen neuen User anlegen, der in diesem Projekt arbeiten soll, machen Sie das mit:

```
#useradd -g projekt -m Projektbenutzer
```

Die Gültigkeit der Benutzerkonten läßt sich auch auf ein ganzes Netzwerk ausweiten. Dazu sollten Sie sich einmal den NIS (Network Information Service) ansehen.

Benutzerrechte vergeben

Sie haben mit NetBSD ein Multi-User-Betriebssystem. Damit das Ganze nicht im Chaos endet, gibt es Beschränkungen für alle Benutzer, die nicht *root* sind. Im einzelnen:

Ein Standardbenutzer hat nur volle Lese-, Schreib- und Ausführungsrechte in seinem Stammverzeichnis, das üblicherweise unter »/home« angelegt wird. Es gibt Dateien, die ein solcher Benutzer ausführen darf. Außerdem gibt es Dateien, die er lesen darf und solche, von denen er die Finger lassen soll. Beispielsweise darf im Verzeichnis »/etc« jeder Benutzer standardmäßig lesen, um zu sehen, wie das System konfiguriert ist, aber niemand außer *root* hat das Recht, diese Daten zu ändern. Ein anderes Beispiel sind die Verzeichnisse »/sbin«, »/usr/sbin« und »/usr/pkg/sbin«, auf diese Verzeichnisse hat nur der Systemverwalter Zugriff.

Die Lese- und Schreibrechte für eine Datei ändert man mit dem Befehl *chmod*. Frei übersetzt bedeutet das »Change Mode« oder zu Deutsche »Wechsle Modus«. Der Befehl sieht so aus:

```
#chmod 755 Datei
```

Regeln können Sie mit diesen Parametern, wer überhaupt was tun darf und wer was darf. Die Erteilung der Rechte geschieht mit Buchstaben oder einer dreistelligen Oktalzahl wie oben geschehen. Die Zahlen bedeuten:

- *4*: Lesen
- *2*: Schreiben
- *1*: Ausführen

Gesetzt dem Fall, daß obige Beispieldatei zu *root* gehört, dann sehen die Rechte so aus:

- *7*: root darf alles mit der Datei machen.
- *5*: Mitglieder seiner Gruppe dürfen die Datei lesen und ausführen.
- *5*: Andere Benutzer habe die gleiche Rechte wie die Mitglieder der Gruppe von *root*.

Die erste Ziffer legt die Rechte des Eigentümers fest; die zweite die Rechte der Gruppe, der der Eigentümer angehört; mit der dritte Zahl wird geregelt, was andere Benutzer damit dürfen. Diese Ziffern werden aufaddiert.

Dann können Sie vor diese Zahl noch eine vierte Ziffer setzen:

- *4*: Die Benutzer-ID wird während der Ausführung gesetzt.
- *2*: Setzen der Gruppen-ID.
- *1*: Das Sticky Bit wird gesetzt.

Ein gesetztes »Sticky Bit« veranlaßt NetBSD, ein einmal geladenes Programm im Arbeitsspeicher zu behalten. Von besonderem Nutzen ist das bei Programmen, die häufig und von mehreren Benutzern gleichzeitig in Gebrauch sind.

Bei Verzeichnissen hingegen ermöglicht das Sticky Bit das Anlegen gemeinsam genutzter Verzeichnisse. Normalerweise muß ein solches Verzeichnis die Rechte *drwxrwxrxw* besitzen.

Jeder kann nun in dieses Verzeichnis schreiben, allerdings auch jede Datei darin löschen. Durch das Setzen des Sticky Bits wird es nur dem Eigentümer einer Datei erlaubt, sie zu löschen.

Das *t*-Recht (Sticky Bit) läßt sich nur vom Superuser vergeben.

u	Eigentümer
g	Gruppe
o	Andere Benutzer
a	Eigentümer, Gruppe und andere Benutzer auf einmal
+	Vergeben von Rechten
-	Entziehen von Rechten
r	Leserecht
w	Schreibrecht
x	Ausführungsrecht von Dateien, auch Durchsuchen von Verzeichnissen und Anlegen von Dateien in Verzeichnissen
s	Setzt Eigentümer- oder Gruppen-ID bei der Ausführung
g	Gruppe
o	Andere Benutzer
t	Sticky-Bit

Tabelle 3.3: Eigentümerrechte

Oktal werden die neuen Rechte durch eine weitere vorangestellte Ziffer ausgedrückt. So werden beispielsweise die Rechte *rwxrwxrwt* durch 1777 ausgedrückt.

3.1.7 root-Passwort vergessen

Auch wenn Sie ausgeloggt sind und aus unerfindlichen Gründen einen geistigen Aussetzer hatten, haben Sie eine Chance, das Passwort für *root* zu ändern und wieder auf Ihre Daten zuzugreifen. Wir gehen an dieser Stelle davon aus, daß noch keine Änderungen an der Konfiguration der Terminals vorgenommen wurde. Machen Sie folgendes:

Starten Sie das System. Nach einer kurzen Zeit erscheint der Bootprompt mit einem Countdown. Sie geben am Prompt ein:

```
#boot netbsd -s
```

Das System fährt in den Single-User-Modus hoch. Anschließend geben Sie

```
#fsck -y
```

ein. Es wird ein Check des Dateisystems ausgeführt. Die nächste Eingabe ist:

```
mount -u /
```

Jetzt wird das Root-Dateisystem gemountet.

Mit

```
fsck -y /usr
```

wird die »/usr«-Partition geprüft, alle Fragen werden automatisch bejaht. *fscheck* kann nur auf nicht gemountete Partitionen ausgeführt werden! Das Mounten des »/usr«-Dateisystems erfolgt mit:

```
mount -u /usr
```

Sollten Sie eine andere Partitionstabelle als die aus der Standardinstallation eingerichtet haben, müssen Sie so auch mit allen anderen Dateisystemen vorgehen. Auf den Check des Dateisystems kann verzichtet werden, wenn man das System vorher sauber heruntergefahren hat.

Wenn Sie damit fertig sind, ändern Sie Ihr *root*-Passwort mit dem Programm *passwd.*

Dann kommt die bereits bekannte Frage nach dem neuen Passwort. Ist das alles erledigt, fährt man das System mit *exit* oder einem Druck auf [Ctrl][D] wieder in den Multi-User-Modus hoch.

3.1.8 Das rc-Startsystem in NetBSD

NetBSD besitzt seit einigen Versionen ein neues Startverfahren. Natürlich wird zuerst der Kernel gestartet, er lädt das Programm *init* nach, das dann die Verwaltung des restlichen Bootvorgangs vornimmt. Bisher ist nichts neues beschrieben – alle Unix-Derivate booten so. Wer Linux kennt, erwartet, daß jetzt über das Init-Programm in ein Runlevel geschaltet wird. Aber nein. Alle BSDs verwenden im Gegensatz zu Linux (Slackware als Ausnahme bestätigt die Regel) anstelle des undurchsichtigen Runlevel-Systems nach System V ein rc-gesteuertes Simple-Init-System.

Dieses Simple-Init-System kennt nur zwei Startmodi: Single-User-Modus und Multi-User-Modus. Der Single-User-Modus läuft ohne Netzwerk und wird nur im Notfall hochgefahren. Dazu wird am Bootprompt die Leertaste gedrückt und dann

```
boot -s
```

angegeben. In allen anderen Fällen startet das System im Multi-User-Modus. Tritt hier ein Fehler auf, beispielsweise weil der Administrator eine Konfigurationsdatei kaputtgespielt hat, läuft das System ebenfalls in den Single-User-Modus. Sind die Reparaturen abgeschlossen, setzt das System den normalen Bootvorgang fort. Dazu wird entweder *exit* oder [Strg] [D] auf der Kommandozeile eingegeben.

Während – um beim Vergleich zu bleiben – bei Linux der Start der grafischen Oberfläche über die Auswahl des entsprechenden Runlevels geschieht, das sich von Distribution zu Distribution auch noch unterscheiden kann, wird bei NetBSD dergleichen über einen Konfigurationseintrag im rc-System entschieden.

Wenn die Benutzeroberfläche nach dem Systemstart eine grafische sein soll, wird das über das *rc-System* geregelt.

Das rc-System umfaßt eine ganze Reihe von Dateien, in denen der Administrator festlegt, welche Dienste beim Systemstart geladen werden. Sie alle haben Auswirkungen, welche Startskripten automatisch geladen werden müssen. Und das ist jetzt neu! Bei FreeBSD und OpenBSD sind grundsätzlich alle Routinen in der Datei »/etc/rc« zusammengefaßt. Die Datei selbst wertet die Konfigurationsdateien »rc.conf«, »rc.local«, »rc.network« und so weiter aus. Bei NetBSD ist die rc-Datei selbst sehr klein. In ihr werden nur die »rc.conf« und »rc.subr« geladen. In der Subroutinen-Datei wird dann angeworfen, was es zu laden gibt. Die zu ladenden Dateien befinden sich – das ist neu in NetBSD – im Verzeichnis »/etc/rc.d« oder in äquivalenten Verzeichnissen, beispielsweise »/usr/pkg/etc/rc.d« oder »/usr/X11R6/etc/rc.d«.

Über das Programm *rcorder* analysiert die rc-Datei, in welcher Reihenfolge die Skripten verarbeitet werden müssen. Nach den Einträgen in der »rc.conf« wird entschieden, welche Daemons benötigt werden.

Diese Art des Starts ist einmalig in NetBSD. Aus diesem Grund ist ihm in diesem Buch das eigene Kapitel 14 gewidmet.

Ein Dienst aus dem rc-System wird nach folgendem Schema gestartet. Der Eintrag

```
start_nfs=NO
```

in der »/etc/defaults/rc.conf« bedeutet, daß NFS (Network File System) nicht gestartet wird.

Wenn rc jetzt den Eintrag

```
start_nfs=YES
```

in »/etc/rc.conf« vorfindet, wird NFS gestartet, weil die Einstellungen des Benutzers in dieser Datei Vorrang vor den Standardwerten haben.

In »/etc/rc.d« gibt es eine Datei mit dem Namen »nfsd«. Im folgenden wird der Inhalt der Datei von NetBSD 1.6 erläutert:

`. /etc/rc.subr`	Diese Datei wird zuerst ausgeführt.
`name="nfsd"`	Der Name des Dienstes.
`rcvar="nfs_server"`	Besagt, daß NFS als Server gestartet werden soll.
`command="/usr/sbin/${name}"`	Das Kommando, das ausgeführt werden soll.
`required_vars="rpcbind"`	Ein zusätzliches Programm, das Voraussetzung für NFS ist. Hier ist es *rpcbind*, früher war das *portmap*.
`load_rc_config $name`	Nach diesem Kommando soll in der »rc.conf« gesucht werden.
`run_rc_command "$1"#!/bin/sh`	Das Kommando soll unter der Shell *sh* ausgeführt werden.

Zur Philosophie von NetBSD gehört es auch, den User nicht übermäßig zu bevormunden. In der »rc.conf« wird des öfteren empfohlen, einige Konfigurationen in besonderen Dateien abzulegen. Das gilt insbesondere für das Routing und die Zuweisung von IP-Adressen zu bestimmten Netzwerkkarten.

Dementsprechend ist es natürlich trotzdem möglich, solche Einrichtungen in der »rc.conf« zu tätigen. Sie funktionieren auch von dort.

Diese »rc.conf« bietet gegenüber den Konzepten diverser Linux-Varianten auch den Vorteil, daß sie weitaus einfacher zu konfigurieren ist und sie längst nicht so unübersichtlich ist wie die einiger Linux-Anbieter, die zu der eigentlichen »rc.conf« gleich noch eine ganze Reihe anderer rc.confs in diversen Unterverzeichnissen anbieten. Das Resultat ist, daß ein Anfänger wie der berühmte Ochs vorm Berge davor sitzt, nach dem Sinn seines Lebens fragt und auf ein in sich geschlossenes System nebst eines zentralen Konfigurationstools angewiesen ist, das dann bei den älteren Versionen auch noch in zwei Versionen auf dem System herumgeistert und dessen Bedienung noch nicht einmal konsistent ist, weil das zweite Produkt auf den Markt muß, obwohl die erste Version noch nicht einmal fertig ist. Der Lerneffekt hinsichtlich einer Konfiguration eines Unix-Betriebssystems wird sich dabei gegen Null bewegen.

Ausprobiert habe ich beide Varianten. Wenn Sie sich hinsichtlich der Syntax in der Konfiguration externer Dateien nicht sicher sind, können Sie diese Dinge auch in der »rc.conf« eintragen, in der das oftmals auch wesentlich einfacher ist.

Die Manpages

Wenn Sie in der Unix-Welt neu sind, sollten Sie außerdem wissen, daß es unter Unix-Betriebssystemen wie NetBSD eine sehr gute Online-Dokumentation gibt, mit der sich andere Betriebssysteme kaum messen können. Das ist das Man-System. Die Manpages von NetBSD stehen unter dem Ruf, die besten überhaupt zu sein. Wenn Sie mal ein paar davon gelesen haben und über eine Möglichkeit verfügen, sie mit denen anderer Systeme zu vergleichen, werden Sie verstehen, warum das so ist. Selbst das kann ein Grund sein, sich NetBSD auf die Maschine zu spielen.

Wie alles unter Unix ist auch dieses System strukturiert, um die Suche nach der passenden Hilfe zu erleichtern. Wenn Sie mehr Grundsätzliches dazu wissen wollen, können Sie das mit diesem Kommando in Erfahrung bringen:

```
#man man
```

Der Aufbau der NetBSD-Dokumentation folgt den unter Unix-Systemen üblichen Standards:

1. Grundlegende Kommandos, Tools und Utilities
2. Systemaufrufe und Fehlernummern
3. C-Bibliotheken
4. Spezielle Dateien und Hardwareunterstützung
5. Dateiformate
6. Spiele
7. Sonstige Informationen
8. Systempflege und die Kommandos, die nur root ausführen darf
9. Interna zum Kernel

Zu jedem Man-Abschnitt gibt es eine Intro-Page, die mit

```
#man 8 intro
```

abgefragt wird.

Wenn Sie bereits einen Drucker konfiguriert haben, können Sie sich diese Seiten auch ausdrucken. Das geht mit:

```
# man man | lpr
```

Interessant ist auch ein Blick in das »rc.d«-Verzeichnis. Schauen Sie sich dort einmal ein paar Dateien an. Wenn Sie schon einmal programmiert haben, werden Ihnen sicher ein paar hochinteressante Einträge auffallen, besonders, wenn Sie eine Linux-Installation besitzen und die Dateien miteinander vergleichen.

3.1.9 Das Netzwerk und seine Basiskonfiguration

Der Einfachheit halber erkläre ich die Grundkonfiguration des Netzwerks in der »alten«, aber noch aktuellen IPv4-Version.

Bevor Sie mit eigenen Experimenten anfangen, an dieser Stelle ein paar Informationen über die Vorgehensweise bei der Vergabe von IP-Adressen. Es gibt für den Privatgebrauch vier Klassifizierungen von Netzwerken. Diese Adressen sind reservierte Adreßräume, die niemals außerhalb des eigenen Netzwerks auftauchen sollten.

Voraussetzungen für den Privatgebrauch

127.0.0.0/8 ist das sogenannte Loopback-Netz, der sogenannte *Loopback-Adreßraum*. Diese Adresse ist nur für rechnerinterne Tests zu verwenden. Diese Adresse ist dem sogenannten Loopback-Interface zugeordnet, das nicht nach draußen kommuniziert. Eine derartige Adresse wird niemals im Internet oder an einer Netzwerkkarte auftauchen. Die Netzwerkadresse ist hier acht Bit lang (127), die Host-Adresse 24 Bit (0.0.1). Die Klassifizierung der Netzwerkadressen erkennt man an der Zahl hinter dem »/«.

10.0.0.0 bis 10.255.255.255: In der Vergangenheit war das der reservierte Bereich des sogenannten »Klasse-A«-Netzes. Heute werden diese Netzwerke als »/8« bezeichnet. Die Netzwerkadresse (10) ist acht Bit lang und die des Hosts dementsprechend 24 Bit. Resultat: Wenige Netzwerke und viele Hosts pro Netzwerk.

172.16.0.0/16 bis 172.31.255.255: Das einst sogenannte »Klasse-B«-Netz. Die aktuelle Bezeichnung ist »/16«. Hier ist die Adresse des Netzes 16 Bit lang und die Host-Adresse ebenfalls 16 Bit.

192.168.0.0 bis 192.168.255.255: Das einstige »Klasse-C«-Netz. Nach den aktuellen Normen ist das ein »/24«-Netz. Die Netzwerkadresse ist 24 Bit lang und die Host-Adresse acht Bit.

Man braucht noch eine Netzmaske, die nach diesem Schema entsteht: Wie Sie ja bereits gesehen haben, besitzen Sie eine Netzwerkadresse, die für den Gebrauch in »richtigen« internen Netzen 8, 16 oder 24 Bit lang sein kann, wenn man von den Loopback-Adressen absieht. Vereinfacht gesehen, sind bei einem /24-Netz also 24 Bits »besetzt«. Diese 24 Bits sind die Netzwerkmaske. Acht Bits entsprechen 256 Adressen, von Null bis 255. Die Netzmaske in einem /24-Netz ist demnach 255.255.255.0. Jetzt ist noch eine sogenannte Broadcast-Adresse erforderlich, um beispielsweise die Zeit weiter verteilen zu können oder netzwerkweite Mitteilungen zu versenden. Die korrekte Adresse ist die mit der höchsten möglichen Nummer im jeweiligen Netzwerk. Wenn Sie ein /24-Netz mit der Netzwerkadresse 192.168.0.0 besitzen, sind Sie mit 192.168.0.255 als Broadcast-Adresse richtig.

Sie brauchen zur Netzwerkkonfiguration eines Rechners ein paar Informationen. Das ist der Typ der Karte, eine IP-Adresse (im heimischen Netz eine /16- oder eine /24-Adresse).

Wenn Sie mit der NetBSD-Maschine im Internet surfen oder von irgendwelchen FTP-Servern Software herunterladen wollen, müssen Sie wissen, welche Adresse der Nameserver Ihres Providers hat, sofern Sie keinen eigenen Nameserver betreiben.

Die Netzwerkmaske ergibt sich aus der IP-Adresse des Rechners. Gleiches gilt für die Netzwerkadresse und die Broadcast-Adresse.

Gesetzt dem Fall, daß Sie eine der beschriebenen Netzwerkkarten besitzen, sollten Sie nun die Datei »rc.conf« bearbeiten. Davon haben Sie zwei: einmal befindet sich eine mit diesem Namen im Verzeichnis »/etc«, eine weitere mit dem gleichen Namen im Verzeichnis »/etc/defaults«. Sie können sich die Datei aus »/etc/defaults« nach »/etc« kopieren, wenn Sie die »/etc/rc.conf« bisher nicht verändert haben. Das geht so:

```
#cp /etc/defaults/rc.conf /etc/
```

Anschließend muß man den Parameter *rc_configured* = *YES* in der »rc.conf« unter »/etc« erneut eintragen.

Dann ermittelt man den Typ der Netzwerkkarte, falls noch nicht bekannt:

```
#dmesg | more
```

Halten Sie Ausschau nach einer Information zu einem Ethernet-Adapter. In unserem Beispielfall handelt es sich um eine billige 100-Mbit-Karte mit einem Realtek 8139-Chipsatz. Der Gerätename, den der Kernel für diese Karte verwendet, ist *rtk0*. *rtk* steht für den Realtek-Chipsatz und die »0« für die »nullte«, also erste Netzwerkkarte im System.

Jetzt suchen Sie nach einem Eintrag mit der Bezeichnung *auto_ifconfig*= und ergänzen ihn mit einem *YES*. Das bedeutet, daß die Netzwerkkonfiguration automatisch durchgeführt wird.

Dann tragen Sie *ifconfig_rtk0*="*192.168.1.7*" (nehmen Sie Ihre IP-Adresse) ein. Wenn Sie über einen Router oder einen Server mit dem Internet verbunden sind, müssen Sie die IP-Adresse dieser Maschine kennen. Diese Maschine ist Ihr »Gateway«, der Weg nach draußen. Die Adresse dieser Maschine tragen Sie beispielsweise so ein:

```
defaultroute="192.168.1.1"
```

Den Rechnernamen tragen Sie unter *hostname* ein:

```
hostname="MeinRechner.Domain.Topleveldomain"
```

Dann müssen Sie den Parameter *auto_ifconfig* auf *YES* setzen.

Anschließend legen Sie eine Datei mit dem Namen »resolv.conf« an, um von der Maschine aus auch andere erreichen zu können. Darin muß folgendes stehen:

```
nameserver 62.225.253.9 # Beispiel. Verwenden Sie bitte den Ihrigen
```

Der erste Testlauf

Um das eben Abgearbeitete zu testen, geben Sie folgendes ein:

```
#ifconfig rtk0 inet 192.168.1.7 netmask 155.255.255.0
```

...und einen Ping, den Sie nach ein paar Paketen mit [Strg][C] wieder abbrechen:

```
#ping 192.168.1.7
```

Etwa so sollte die Antwort aussehen:

```
PING 192.168.1.7 (192.168.1.7) from 192.168.1.1 : 56(84) bytes of data.
64 bytes from router.lindloff.com (192.168.1.7): icmp_seq=0 ttl=255 time=585 usec
64 bytes from router.lindloff.com (192.168.1.7): icmp_seq=1 ttl=255 time=649 usec
64 bytes from router.lindloff.com (192.168.1.7): icmp_seq=2 ttl=255 time=548 usec
64 bytes from router.lindloff.com (192.168.1.7): icmp_seq=3 ttl=255 time=562 usec

--- 192.168.1.5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.548/0.561/0.585/0.014 ms
```

Wenn die Antwort, die Sie bekommen, so aussieht wie diese hier, ist der Rechner bereits ein »fertiger« Netzwerkrechner, dem noch ein paar Features fehlen, die in den nächsten Kapiteln besprochen werden.

Unter NetBSD haben Sie noch eine weitere Möglichkeit, als diese Daten in der »rc.conf« zu speichern. Sie können Ihre Konfigurationsdaten auch, ähnlich wie in den klassischen Unix-Varianten, in einzelnen Dateien unter »/etc« ablegen. Genaugenommen wird in einigen Dokumentationen sogar empfohlen, die »rc.conf« in Ruhe zu lassen und den alternativen Weg zu gehen. Die Variante mit der »rc.conf« funktioniert aber genauso gut und ist weitaus einfacher zu verwalten, wenn das System klein und diese Datei noch übersichtlich ist.

Für die Definition der Route kommt die Datei »/etc/mygate« zum Einsatz. Man trägt die IP-Adresse des Routers ein:

```
192.168.1.1
```

Dann braucht man eine Datei für jede einzelne Netzwerkkarte. In unserem Fall ist das *rtk0*. Sie heißt »ifconfig.rtk0«. Folgendes muß in der Datei stehen:

```
inet 192.168.1.2 netmask 255.255.255.0
```

NE2000-Netzwerkkarten und ihre Nachfolger

Mit NetBSD funktionieren nahezu alle Netzwerkkarten. Ist das bei Ihnen nicht der Fall oder ist Ihre Karte ein neueres ISA-Modell, kann es sein, daß Sie den Plug&Play-Modus im BIOS des Rechners abschalten müssen oder daß Interrupt und I/O-Port nicht korrekt gesetzt sind.

PCI-Modelle:

Im PCI-Bus funktionieren solchen Karten immer.

3Com: Ziemlich alle Modelle. Die Einstellungen der Karte müssen nicht geändert werden.

Intel: Gelten als die am besten unterstützten Karten für BSD-Systeme. Wenn Sie eine solche Karte besitzen: Gratulation!

Realtek: Diese Karten funktionieren im Regelfall zuverlässig und sind oft preiswert zu bekommen. Genaugenommen besitzen nahezu alle preiswerten bis billigen 100-MBit-Karten für den PCI-Bus diesen Chipsatz. Nachteil dieser Karten: Sie funktionieren, aber sie sind im Vergleich zu den oben genannten manchmal recht langsam, weil die Fertigungsqualität der Chips ungleichmäßig ist.

Alle anderen: Diese Karten sind nur die am weitesten verbreiteten Modelle auf dem Markt. Wenn Sie eine andere Karte als die beschriebenen Modelle verwenden, wird sie im Regelfall auch funktionieren. Durchsuchen Sie die Kernelmeldungen nach einem Netzwerkgerät.

ISA-Karten:

Diese Karten für alte Rechner (moderne Maschinen haben ohnehin keine ISA-Steckplätze mehr) funktionieren im Regelfall, wenn sie nicht auf den Plug&Play-Modus gesetzt und auf einen Interrupt von 10 und den I/O-Port 300 konfiguriert sind. Sollte das nicht der Fall sein, müssen Sie entweder auf die Installation aus dem Netzwerk verzichten oder die Karte vor der Installation von NetBSD umkonfigurieren. Dafür gibt es ein paar DOS-Tools, die mit der Karte geliefert werden. Wenn Sie diese Tools nicht haben, eventuell weil die Diskette verlorengegangen ist, könnten Sie zumindest bei einigen Herstellern Glück haben, daß zufälligerweise die Daten so gesetzt sind, daß die Standardwerte, die NetBSD erwartet, verwendet werden. Das ist aber leider nicht immer der Fall. Eine solche »jumperless« ISA-Karte hat sich bei meinem NetBSD-basieren DSL-Router trotz statisch gesetzter Interrupts und I/O-Adressen mal als *ne0*, beim nächsten Boot dann als *ne2* und dann mal als *ne3* ausgegeben. Das Ergebnis können Sie sich sicher vorstellen: Mal geht's und mal nicht. Das heißt, daß Sie das so machen können, aber irgendwann werden Sie zu müde dafür.

Wenn Sie noch eine von den richtig alten Karten mit den Jumpern besitzen, sollten Sie keine Probleme damit haben. Wenn Sie eine jumperless Karte besitzen, sich diese Karte sich wie beschrieben verhält und ein paar Euro übrig haben, sollten Sie diese wirklich durch einen PCI-Nervenschoner oder eine zuverlässige, richtig alte ISA-Karte vom Flohmarkt ersetzen.

Ist man über einen Router mit dem Internet verbunden, muß man außerdem noch einen Nameserver definieren. In der »/etc/resolv.conf« geschieht dies mit einem Eintrag wie diesem:

```
nameserver 62.225.253.9
```

Das Beispiel zeigt den DNS-Proyxserver für ADSL-Verbindungen via T-Online im Raum Hannover. Sollten Sie die IP-Adresse Ihres Nameservers nicht kennen, wird Ihr Anbieter Ihnen sicherlich gerne weiterhelfen.

Wenn Sie bis hierher gekommen sind, sollten Sie das System jetzt einmal neu starten, um zu sehen, ob alles neu Konfigurierte auch so funktioniert, wie Sie sich das gedacht haben.

3.1.10 Software nachinstallieren

Es gibt zwei Möglichkeiten, Software aus der Package-Kollektion von NetBSD nachzuinstallieren. Einmal von einer CD aus, alternativ kann man sich das notwendige Material vom FTP-Server von NetBSD oder einem seiner Mirrors besorgen.

Beide Verfahren haben ihre Vor- und Nachteile.

Die Installation von der CD

Die Installation von der CD aus ist recht einfach:

- Mounten Sie die CD.
- Führen Sie *pkg_add {paketname}.** aus.

Wenn Sie das Paket wieder loswerden wollen, geben Sie *pkg_delete {paketname}* ein. Wollen Sie wissen, welche Pakete auf ihrem System installiert sind, führen Sie *pkg_info* aus.

Diese Vorgänge darf nur *root* ausführen.

Manche Pakete brauchen ein paar andere, um zu funktionieren. Der im Kapitel 13 zum Drucken beschriebene Magicfilter ist beispielsweise so eine Paket, das, sofern es nicht im Zusammenhang mit einem anderen Paket bereits installiert ist, nachinstalliert wird.

Installation aus den NetBSD-Quelldaten

Die zweite Möglichkeit, die nur bei einer schnellen Internetverbindung zu empfehlen ist, ist, sich aus dem Internet nur ein Paket oder eine Sammlung zu beschaffen und diese zu installieren. Dazu gibt es eine Datei namens »pkgsrc.tar.gz«, die Sie auf dem NetBSD-Server oder Ihrem Lieblingsmirror finden können. Der URL zum NetBSD-Server ist *ftp://ftp.netbsd.org/pub/NetBSD/packages/pkgsrc.tar.gz*.

Speichern Sie diese Datei unter dem Verzeichnis »/usr« und entpacken Sie sie mit dem Befehl

```
#tar -xvzf pkgsrc.tar.gz
```

Das Auspacken dauert eine ganze Weile.

Nach dem Auspacken wechseln Sie in das Verzeichnis »pkgsrc«. Sie werden dort einige Ordner vorfinden, deren Bezeichnungen selbsterklärend sind.

Exerzieren wir ein Beispiel durch. Wechseln Sie in das Verzeichnis »Editors« und installieren Sie den Editor »joe«. Das geht folgendermaßen:

```
#cd joe
#make install
```

Jetzt wird das Paket mit dem Namen »joe« vom entsprechenden Server heruntergeladen, ausgepackt, kompiliert und anschließend installiert. Wenn Sie das Programm wieder loswerden möchten, machen Sie das im selben Verzeichnis mit *make uninstall*.

3.2 Das Programm sushi

von Jörg Braun

Mit der Version 1.6 von NetBSD wurde das Verwaltungsprogramm *sushi* für die grafisch orientierte Software-Verwaltung eingeführt. Das Programm selbst befindet sich in »/usr/sbin« und ist damit dem Systemadministrator vorbehalten. Die Konfigurationsdateien befinden sich in »/usr/share/sushi«.

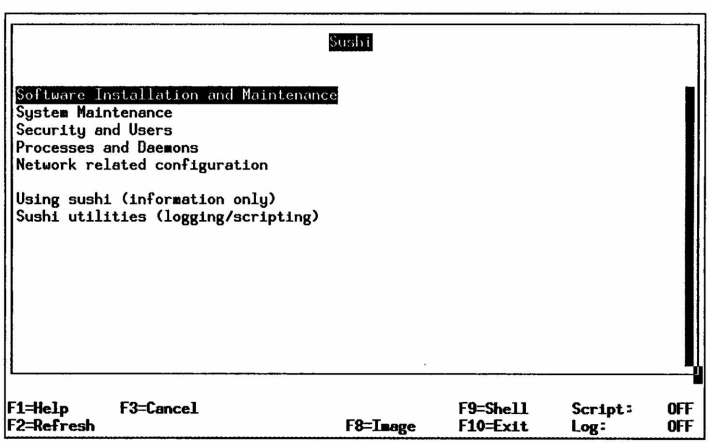


Abb. 3.2: Der Hauptbildschirm von sushi

sushi lädt beim Start die Dateien in diesem Verzeichnis und zeigt den Hauptbildschirm (Abb. 3.2) an.

Zur Verfügung stehen Menüs zur Installation und Verwaltung von Software-Packages, zur Betriebssystem-Verwaltung, zur Systemsicherheit und Benutzerverwaltung, zur Anzeige der Daemons und zum Netzwerk. Gegenüber der Verwendung der entsprechenden kommandozeilenorientierten Programme stellt *sushi* für den weniger erfahrenen Anwender eine gewisse Erleichterung dar. Allerdings sollte man sich von *sushi* noch nicht allzu viel erwarten. Diese erste Version wird aber in zukünftigen Releases von NetBSD sicherlich deutlich ausgebaut, insbesondere deshalb, weil es sehr einfach zu erweitern und auch zu lokalisieren ist.

Die Grundidee des Aufbaus von sushi ist verzeichnisorientiert. Das Hauptverzeichnis muß nur eine Datei »index« enthalten. In ihr sind weitere Menüs und deren Funktion als Klartext enthalten. Hier der Inhalt der Grund-Indexdatei:

```
# $NetBSD: index,v 1.5 2001/04/26 02:26:16 garbled Exp $
install         install Software Installation and Maintenance
system          system  System Maintenance
users           users   Security and Users
procs           procs   Processes and Daemons
network         network Network related configuration
BLANK           BLANK   BLANK
info            info    Using sushi (information only)
util            util    Sushi utilities (logging/scripting)
```

Übersetzt man diese Datei und legt unter dem Namen »index.de« die Datei mit den folgenden Übersetzungen ab,

```
# $NetBSD: index,v 1.5 2001/04/26 02:26:16 garbled Exp $
install         install Software-Installation und Verwaltung
system          system  Systemverwaltung
users           users   Sicherheit und Benutzerverwaltung
procs           procs   Prozesse und Daemons
network         network Netzwerk-Konfiguration
BLANK           BLANK   BLANK
info            info    Hilfe zu sushi (Nur zur Information)
util            util    Sushi-Tools (Logging und Skripting)
```

wird das Menü beim nächsten Aufruf auf einem Rechner, in dessen Umgebung *LANG=de* definiert ist, in deutscher Sprache angezeigt (Abb. 3.3). Der primäre Aufbau der sushi-Verzeichnisse ist damit klar: gesteuert wird über die Index-Datei. Zu jedem Eintrag in der Index-Datei (sofern er nicht »BLANK« heißt) gehört jetzt ein Verzeichnis, zum Topic »install« also das gleichnamige Verzeichnis. Es enthält wieder eine Datei »index«, die dann wieder auf mehrere Unterverzeichnisse verweist. Betrachtet man die Inhalte der hier enthaltenen Unterverzeichnisses »install« oder »installb« an, sieht man, daß hier keine weitere Datei »index« enthalten ist.

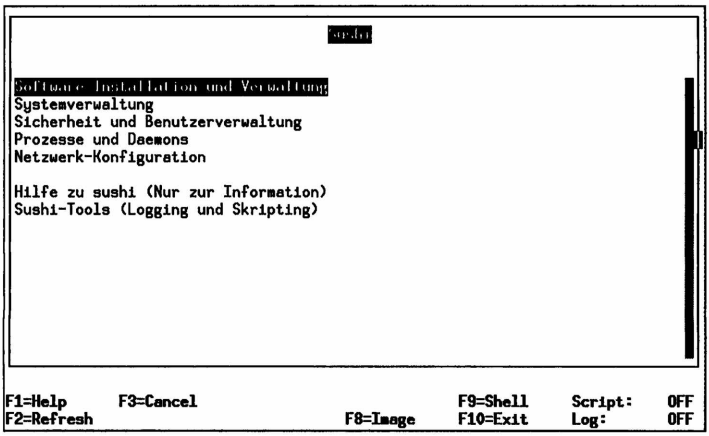


Abb. 3.3: sushi ist sehr flexibel. Die Übersetzung des Hauptmenüs zeigt sofort Wirkung

Die hier enthaltene Datei »form« enthält die folgenden Zeilen:

```
# $NetBSD: form,v 1.2 2001/01/24 09:34:37 garbled Exp $
script:script1,@@@1@@@  Install which package?
noedit:@@@1@@@          Selected Category:
```

Die zusätzliche Datei »preform« enthält die Zeilen

```
# $NetBSD: preform,v 1.1 2001/01/24 09:35:57 garbled Exp $
script:script2          Select a package category
```

Hier werden also Aktionen durchgeführt. Das Verzeichnis enthält in der Tat auch die in den Dateien erwähnten ausführbaren Skripte »script«, »script1« und »script2«.

3.2.1 Software-Verwaltung

Ruft man das Menü zur Software-Verwaltung und Neuinstallation auf, stößt man aber sofort an die Grenzen des Systems.

Es wird sofort versucht, sich auf dem NetBSD-FTP-Server einzuloggen, um nach den dort befindlichen Paketen zu suchen. Ein Auswahl – vergleichbar mit dem Programm */stand/sysinstall* von FreeBSD – fehlt. Wenig erfreulich ist auch, daß nun gleich noch die Konsole hängt, weil sushi nicht aufgibt, einen eventuell nicht erreichbaren Server zu finden. Das Programm läßt sich nur noch von einer anderen Konsole aus mit einem *kill -9* abbrechen.

Die in der Datei »script« befindliche Abfrage wird offensichtlich nicht angesprungen:

```
if [ -z "$2" ]; then
   echo "There was an error finding your package source directory."
   echo "Please set the environment variable \$PKGSRCDIR to the location"
   echo "of your package sources."
   exit 1
fi
```

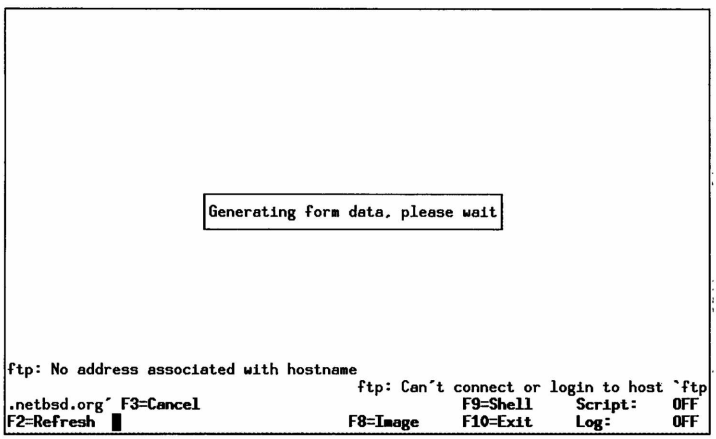


Abb. 3.4: Noch nicht so ausgereift!

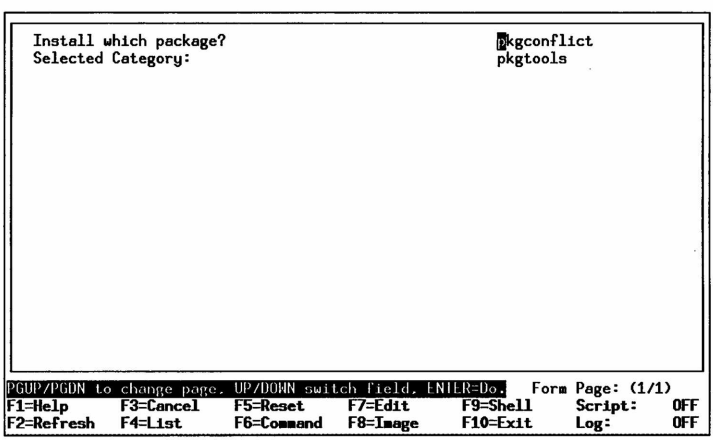


Abb. 3.5: Die Bewegung durch den Pkgsrc-Tree ist wenig komfortabel gelöst

Aber man lernt daraus, daß man die Variable *PKGSRC* auf jeden Fall setzen sollte! Ist der Package-Source-Tree installiert, erlaubt *sushi* auch die Installation aus den Quellen.

Allerdings ist die Auswahl relativ mühsam. Es ist kein bildschirmorientiertes Browsen möglich. Man muß sich mit der Tabulatortaste durch die Verzeichnisse und dann durch die Unterverzeichnisse bewegen.

Die Auswahl eines Pakets funktioniert. Leider wird aber nicht überprüft, ob ein ausgewähltes Paket bereits installiert ist.

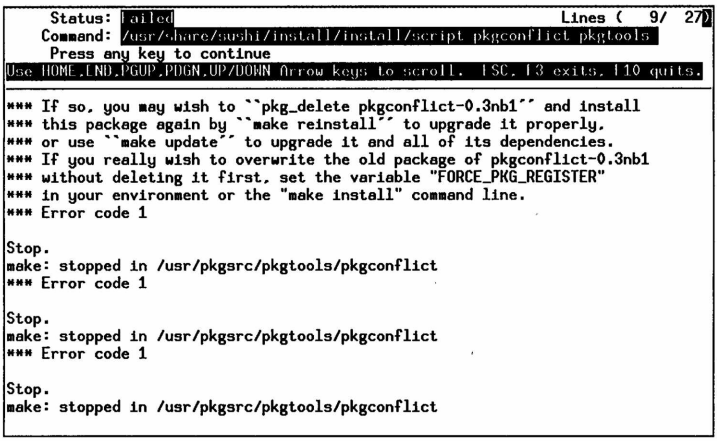


Abb. 3.6: Eigentlich hätte das Programm vorher prüfen müssen, ob das selektierte Paket bereits installiert ist

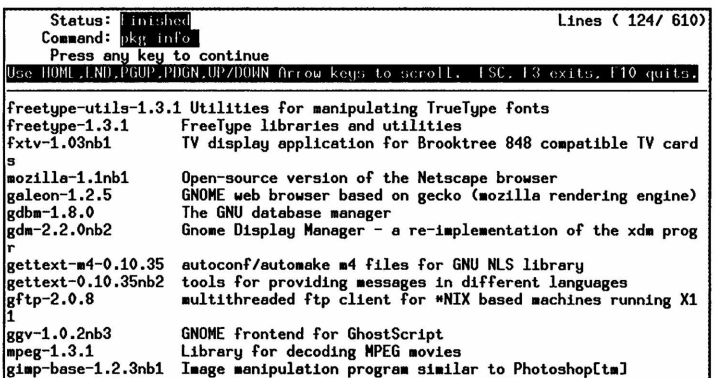


Abb. 3.7: Die Sortierung der installierten Pakete nach dem Installationszeitpunkt ist wohl ziemlich daneben!

So bleibt nur, vor dem Aufruf der Installation eines Pakets selbst mit dem Menüpunkt *List installed Packages* auf die Suche zu gehen. Allerdings hat sich der Autor von *sushi* bei diesem Menüpunkt auch nicht mit Ruhm bekleckert. Die Sortierung der installierten Pakete erfolgt nicht alphabetisch, sondern nach dem Datum der Installation. Gerade diese Art der Sortierung ist aber vollkommen unerheblich und kostet sehr viel Zeit, wenn man überprüfen will, ob ein Programm bereits installiert ist.

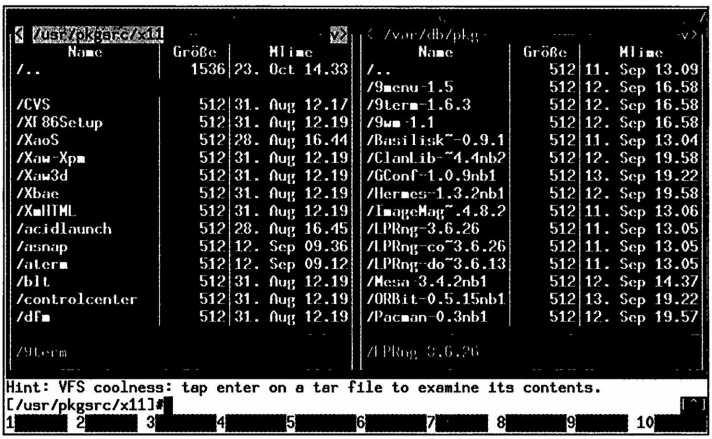


Abb. 3.8: Immer noch sinnvoller als die Paketverwaltung mit sushi ist die Übersicht im Midnight Commander

Als Werkzeug für die Software-Installation ist sushi also nur äußerst bedingt geeignet. Da ist es wirklich besser, den Midnight Commander zu installieren und damit durch die Verzeichnisse zu gehen.

3.2.2 Systemverwaltung

Auch wenn die Verwaltung der Packages wenig gut gelungen ist und von ihrem Gebrauch eigentlich nur abgeraten werden kann, hat das Programm doch auch seine guten Seiten. Die Systemverwaltung funktioniert besser.

Offensichtlich hat der Autor von sushi hier mehr Mühe aufgebracht. Die Verwaltung der Daemons ist – wie Abb. 3.10 zeigt – recht gut gelungen. Auch das Scrollen durch die Einträge und die Änderungsmöglichkeiten machen einen passablen Eindruck. Gegenüber der manuellen Modifikation hat man den Vorteil, nicht nur eine Variable anzeigt zu bekommen, deren Funktion man erraten muß. Es wird für jeden Eintrag eine ausführliche Beschreibung angezeigt.

Auch der nächste Menüpunkt für die Anpassung der Konfiguration des Internet-Daemons (»/etc/inetd.conf«) funktioniert. Allerdings besteht hier das Problem, daß nur die vorhandenen Einträge angezeigt werden. Neue einzufügen ist nicht möglich.

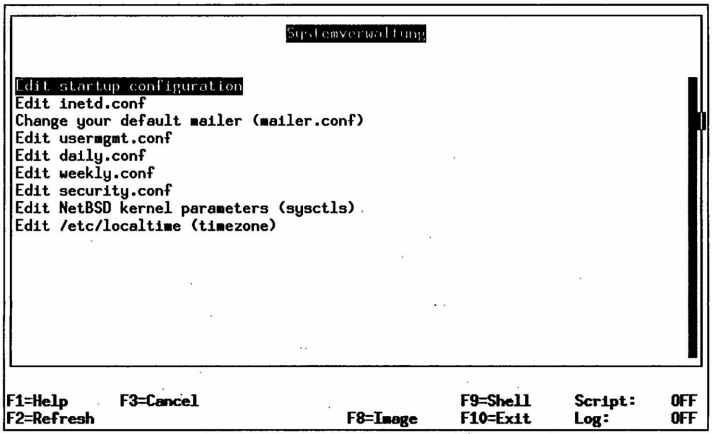


Abb. 3.9: Das Hauptmenü zur Systemverwaltung

Auch die restlichen Punkte des Systemverwaltungsmenüs funktionieren mehr oder minder.

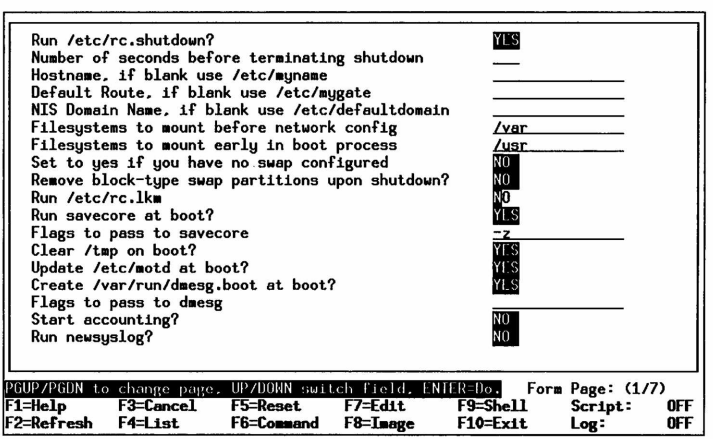


Abb. 3.10: Die Modifikation der Startkonfiguration ändert die Datei »/etc/rc.conf«

Mit *Change you default mailer (mailer.conf)* wählt man einen Mailer-Daemon aus, *Edit usermgmt.conf* funktioniert nicht und führt zum Abbruch des Programms, wenn die Datei »usermgnt.conf« fehlt.

Mit *Edit daily.conf*, *Edit weekly.conf* und *Edit security.conf* werden die entsprechenden Dateien in »/etc« bearbeitet wie schon für die »rc.conf« und die »inetd.conf« vorgestellt.

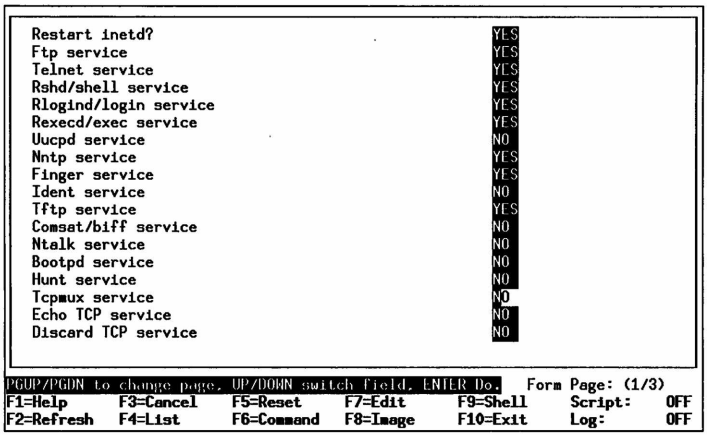


Abb. 3.11: Die Konfiguration des Internet-Daemons

Interessant liest sich auf den ersten Blick der Menüpunkt *Edit NetBSD kernel parameters (sysctls)*. Leider fehlt hier aber auch eine komfortable Browsing-Möglichkeit. Wie bei den Verzeichnissen des Package-Source-Trees muß man sich mit der Tabulatortaste durch die einzelnen Einträge klicken, was zwar funktioniert, aber insgesamt doch sehr mühselig ist.

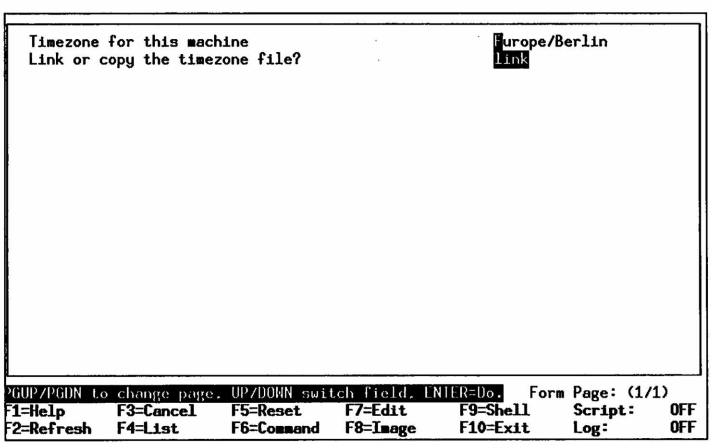


Abb. 3.12: Die Auswahl der Zeitzonendatei ist so wenig praktisch

Edit /etc/localtime (timezone) funktioniert auch, ist aber deutlich weniger komfortabel als im Grund-Systeminstallationsprogramm gelungen. Allerdings wird man diesen Menüpunkt wohl auch nur sehr selten benötigen.

Auch hier muß man sich mit [Tab] durch die Einträge – in diesem Fall die vorhandenen Zeitzonendateien – bewegen. Die Übersicht leidet darunter. Hinzu kommt, daß es nicht möglich ist, sich rückwärts durch die Liste zu bewegen. Hat man sich vertan und zu schnell gewechselt, muß man mit [F3] das Menü verlassen und neu starten. [Alt]+[Tab] funktioniert genauso wenig wie [Shift]+[Tab] und [Ctrl]+[Tab] verläßt analog zu [F3] das Menü.

3.2.3 Sicherheit und Benutzer

Eigentlich klingt der Name dieses Menüs vielversprechend. Allerdings verspricht er mehr als er (bei NetBSD 1.6) halten kann.

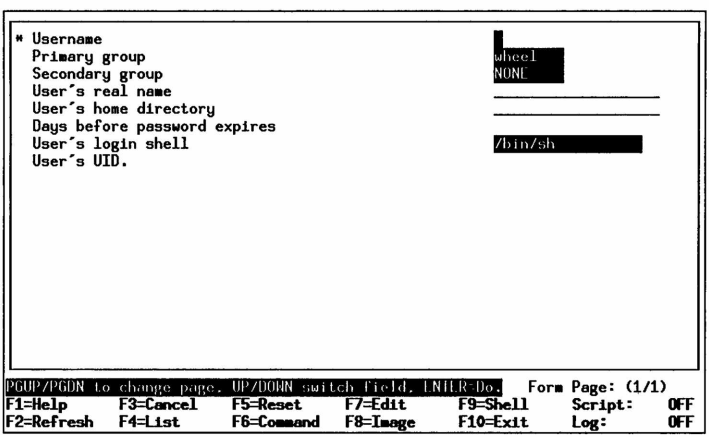


Abb. 3.13: Die Maske für das Anlegen eines neuen Users in sushi

Das Menü enthält nur die Einträge

- List users on this System,
- Add a new user to the system,
- Delete a user on the system und
- Disable a user on the system

Viel mehr gibt es auch nicht zu sagen. *List users ...* fördert den Inhalt der Benutzerdatenbank auf den Bildschirm, die anderen Menüpunkte sich recht praktisch, da es erheblich bequemer ist, einen neuen Benutzer in einer Bildschirmmaske als mit der langen Kommandozeile von *useradd* anzulegen.

Was hier fehlt sind Auswahlboxen, beispielsweise welche User bereits mit welchen IDs angelegt sind, eine Vorauswahl der Shells und vieles mehr.

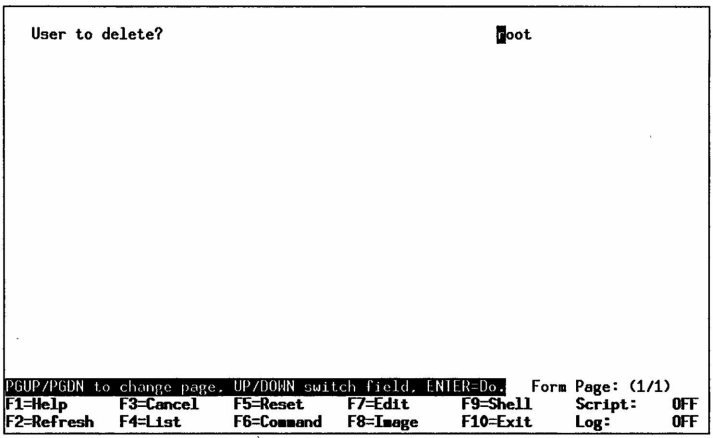


Abb. 3.14: Soll man oder soll man nicht?

Ganz hanebüchen wird es aber, wenn man den Menüpunkt zum Entfernen eines Users vom System aufruft. Als erste Auswahl wird ausgerechnet der *root*-Account angeboten! Wer schon einmal *root* von einer Unix-Maschine gelöscht hat, weiß was das bedeutet.

Daß hier der Root-Account an erster Stelle angeboten wird, ist eine grobe Fahrlässigkeit. Das gleiche gilt auch für den Menüpunkt *Disable*. Da hier ebenfalls nach der User-ID sortiert wird, ist an erster Stelle der Administrator-Account zu sehen.

3.2.4 Prozesse und Daemonen in sushi

Das Verwaltungstool bietet als vorletzten Hauptmenüpunkt die Möglichkeit, Prozesse anzusehen und Daemons an- und abzuschalten. Die Menüpunkte sind:

- Show configured and running daemons
- Show all daemons
- Start configured daemons
- Start any daemon
- Stop configured daemons
- Stop any daemon
- Restart configured daemons
- Restart any daemon
- Show top 15 processes by cpu usade
- Show top 15 processes by memory usage
- Show top 15 processes by cpu time

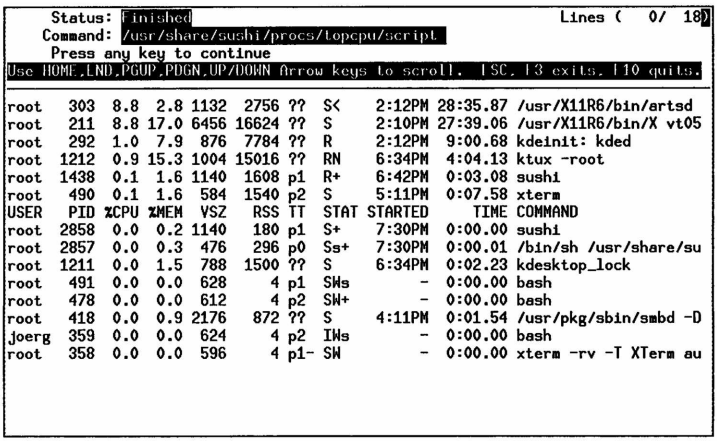


Abb. 3.15: Die Anzeige der fünfzehn die CPU am meisten fordernden Prozesse

Die Übersichten der Daemons sind recht gut gelungen, auch die Punkte zum Starten, Anhalten und Neustarten funktionieren. Wie bei den anderen Auswahlen ist die Art der Auswahl mit der [Tab]-Taste aber nicht besonders benutzerfreundlich.

Die Anzeige der Prozesse ist ebenfalls nicht schlecht gelöst. Ein Aufruf des Programms *top* sieht aber genauso aus und hierbei wird auch noch ständig die Anzeige aktualisiert.

3.2.5 Netzwerk-Konfiguration

Die Netzwerk-Konfiguration ist bei NetBSD recht umständlich gelöst, da keine Werkzeuge dafür zur Verfügung stehen. Man muß also wissen, wie es funktioniert, um zum Erfolg zu kommen. *sushi* ist in dieser Hinsicht schon eine gute Hilfe. Das Netzwerk-Menü besitzt die Einträge

- Show or configure network adapters
- Modify the hosts file
- Show or configure basic routing
- Modify the machine's hostname
- Modify NFS exported filesystems
- Setup or Configure NIS/YP

Die gute Nachricht: Alle Menüpunkte funktionieren und zwar (wenn auch mit kleinen Schwächen) so wie erwartet.

Daß die Adresse statisch angezeigt wird, ist eigentlich nicht korrekt, da die IP-Adresse vom DHCP-Server geholt wird. Wer aber einen DHCP-Server laufen hat, wird sich um die Konfiguration der IP-Adressen ohnehin kaum kümmern und das Menü nicht verwenden. Beim Einsatz statischer IP-Adressen läßt das Menü aber keine Wünsche offen.

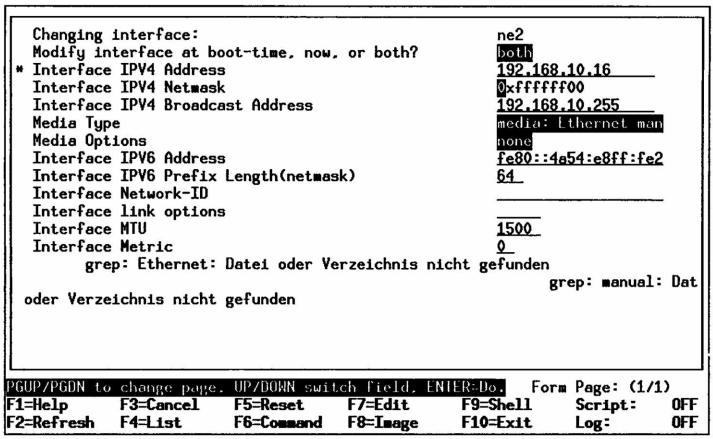


Abb. 3.16: Nicht ganz fehlerfrei dargestellt, aber es funktioniert

Vielleicht sollte man von einem Programm, das so neu ist wie *sushi*, nicht allzuviel erwarten. Es wird ein ausbaufähiger Ansatz gezeigt, der allerdings noch zahlreiche Kinderkrankheiten besitzt. Daß beim Löschen der User an erster Stelle der Name *root* steht, ist heikel. Es wurde auch nicht ausprobiert, was passiert, wenn man an dieser Stelle die Eingabetaste betätigt.

Was an sushi stört, ist der mangelnde Komfort bei der Auswahl aus Listen. Ein Listen-Kontrollelement fehlt und macht die Arbeit umständlicher als nötig.

Was mir persönlich nicht ganz klar ist, ist, warum dieses Programm mit der Version 1.6 von NetBSD in den offiziellen Verzeichnisbaum des Betriebssystems aufgenommen wurde. Eine Verlagerung in das Paketsystem wäre bei einem Betriebssystem, wo man sich als konservativ versteht und wo man Wert darauf legt, daß nur sichere und gut getestete Software aufgenommen wird, eher zu verstehen gewesen.

Dem sonstigen allgemeinen Niveau von NetBSD entspricht dieses Programm jedenfalls nicht. Es kann zwar eingesetzt werden und einige Menüs haben durchaus mittleren bis größeren Nutzwert. Die instabile Bedienung und die Gefahren, die aus der Fehlbedienung entstehen können, müssen dabei aber ständig im Hinterkopf behalten werden.

3.3 Datenträgerverwaltung auf dem i386

von Jörg Braun

Die Verwaltung der Festplatten und auch anderer Datenträger auf dem PC ist eine Angelegenheit für sich. Ursprünglich waren die Festplatten recht klein und die Auswahl an Betriebssystemen war nicht groß. So wurde eine einfache Möglichkeit gefunden, Festplatten einzuteilen und diese Einteilung zu verwalten. Die Ableitung des Aufbaus einer Festplatte erfolgte dann konsequent von der der MS-DOS-Diskette. Da man mehrere Verwaltungseinheiten auf einer Festplatte unterbringen wollte, wurde zum

Bootsektor der Diskette noch ein sogenannter Master Boot Record definiert, der dem Bootsektor übergeordnet wurde. Dabei passiert folgendes:

Nachdem das System eingeschaltet ist, lädt der BIOS-Bootstrap den Sektor 1, Spur 0 des Start-Datenträgers an die Speicherstelle 0000h:7C00h. Wenn die Position 0000h:7DFEh, also die letzten zwei Byte dieses Sektors, den Wert AA55h besitzen, übergibt die BIOS-Laderoutine die Kontrolle an die Speicherstelle 0000h:7C00h. Falls der Wert an dieser Adresse ein anderer ist, handelt es sich um keinen gültigen Startcode und die BIOS-Routine hält das System an.

Bei Disketten ist an dieser Stelle der Bootvorgang soweit beendet, als daß nun vom Bootsektor entweder die berühmte Fehlermeldung ausgegeben wird, daß die Diskette nicht startfähig ist, oder, falls alles in Ordnung war, wird vom Bootsektor das Betriebssystem gestartet.

Bei Festplatten befindet sich an dieser hardkodierten Position Sektor 1, Spur 0 nicht der Bootsektor, sondern der Master Boot Record mit der Partitionstabelle. Wenn dieser Sektor in den Speicher an die Adresse 0000h:7C000h geladen wird, reloziert er sich mit einem anderen Sektor, normalerweise mit dem Sektor der als aktiv vermerkten Partition. Diesen lädt er also an seine eigene Adresse 000h:7C00h. Dadurch wird dann dieser Sektor genauso ausgeführt wie der Bootsektor einer Diskette.

Dieser Master Boot Record umfaßt genau einen Sektor der Festplatte. Viel Platz ist also nicht, schließlich müssen die Daten der Festplattenaufteilung und der Bootcode für das Laden der Startpartition untergebracht sein.

Als man die Festplattenaufteilung ins Leben rief, schienen vier Partitionen pro Festplatte ausreichend. Damit wurde im Master Boot Record auch genau Platz für diese vier Partitionen festgelegt. Der jeweilige Aufbau ist in Tabelle 3.4 gezeigt.

Wert	Beschreibung
Byte (8 Bit)	Aktiv-Flag (Aktiv = 80h, nicht aktiv = 0)
Byte (8 Bit)	Kopf, auf dem die Partition beginnt
Byte (8 Bit)	Erster belegter Sektor
Byte (8 Bit)	Erster belegter Zylinder
Byte (8 Bit)	System-ID (Kennung der Partition)
Byte (8 Bit)	Kopf, auf dem die Partition endet
Byte (8 Bit)	Letzter belegter Sektor
Byte (8 Bit)	Letzter belegter Zylinder
DWord (32 Bit)	Startadresse in logischen Sektoren
DWord (32 Bit)	Größe der Partition in logischen Sektoren (MByte = Wert / 2048)

Tabelle 3.4: Der Aufbau eines Eintrags im Master Boot Record

Beim Erscheinen von MS-DOS 3.3 waren die Festplatten dann schon etwas größer, die maximale Partitionsgröße bei DOS war zu dieser Zeit aber auf 32 MByte festgelegt. Nun kann MS-DOS aber nicht mehrere parallele DOS-Boot-Partitionen verwalten, also mußte man etwas erfinden, was diese Grenze aufhebt. Dazu wurde das »Laufwerk in der erweiterten Partition« ersonnen.

Eine erweiterte Partition ist selbst nichts anderes als eine primäre Partition im Master Boot Record. Sie besitzt nur eine andere Kennung. Sie dient aber als Container für weitere Partitionen, die eben diese zusätzlichen DOS-Laufwerke darstellen.

Dabei ist im Bootsektor jedes Laufwerks in der erweiterten Partition der Startpunkt des nächsten Laufwerks vermerkt. Ist der Bereich mit Nullen gefüllt, bedeutet das, daß kein Laufwerk mehr folgt.

Mit dem Wissen um diese Aufteilung der Festplatte versteht man, wie Programme arbeiten, mit denen die Festplatte aufgeteilt wird. Hier war bisher auch nur die Rede davon, was maximal möglich ist. Eine Festplatte braucht selbstverständlich nicht so viele Partitionen enthalten, außerdem muß eine erweiterte Partition nicht die vierte sein. Es ist genauso möglich, eine normale primäre Partition anzulegen und den Rest der Festplatte komplett mit einer erweiterten Partition zu füllen, die dann die Nummer 1 (da ja die Zählung bei Null beginnt) besitzt.

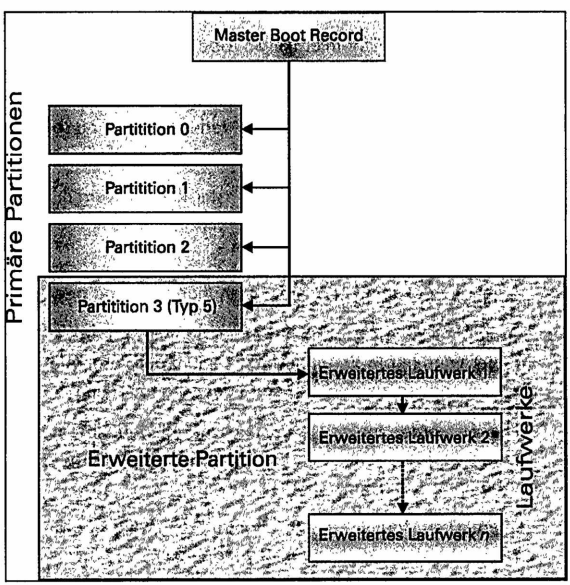


Abb. 3.17: Die Verkettung primärer Partitionen und der Laufwerke in der erweiterten Partition. Unter DOS/Windows dürfen der erweiterten Partition keine anderen mehr folgen, Linux ist großzügiger. Die Reihenfolge der Einträge im Master Boot Record braucht außerdem nicht mit der physikalischen Reihenfolge auf der Festplatte übereinzustimmen

Kommando-zeilen-parameter	Funktion
-0	Aktualisierte nicht-interaktive Partition Slot (Slice) 0.
-1	Aktualisierte nicht-interaktive Partition Slot (Slice) 1.
-2	Aktualisierte nicht-interaktivePartition Slot (Slice) 2.
-3	Aktualisierte nicht-interaktive Partition Slot (Slice) 3.
-a	Wechseln der aktiven (der Boot-) Partition. Im interaktiven Modus erfolgt die Abfrage automatisch am Schluß der Arbeiten.
-b cyl/hd/sect	Angabe der BIOS-Parameter für Zylinder, Köpfe und Sektoren. Dies ist ein Zusatzparameter für *-u*.
-B	Installieren oder Aktualisieren des Bootselektor-Codes auf i386.
-c bootcode	Angabe, aus welcher Datei der neue Bootcode für den Master Boot Record gelesen werden soll. Auf PCs ist die Voreinstellung »/usr/mdec/mbr«, bei anderen Geräten wird der Bootcode leergelassen.
-f	Startet *fdisk* im nicht-interaktiven Modus. Die Plattenparameter können nur über *-b* angegeben werden. Dieser Parameter ist dazu gedacht, daß *fdisk* als Backend für Skripten eingesetzt werden kann. Die gleichzeitige Verwendung von *-f* und *-u* verhindet die Eingabe der Start- und Endadresse des Slices. Sie werden aus der BIOS-Geometrie ermittelt.
-i	Explizite Neuinitialisierung des Master-Records vergleichbar zum Aufruf von *fdisk /mbr* unter MS-DOS. Die Partitionstabelle selbst wird dabei nicht verändert, nur der Bootcode.
-l	Listet alle bekannten Partitionskennungen auf und beendet.
-s id/start/size	Angabe der Partitionsdaten (Kennung, Start und Größe). Dieser Parameter kann nur in Verbindung mit einem der Parameter -0, -1, -2 oder -3 angegeben werden und dient der automatischen Festlegung.
-S	Hiermit werden einige Umgebungsvariablen für sh-Befehle gesetzt. Im Normalbetrieb ist dieser Parameter keine Bedeutung. Er ist vom alten Installationsskript verwendet.
-u	Interaktiver Modus. Jede Partition auf der Platte wird angezeigt, dann wird gefragt, ob sie bearbeitet werden soll.

Tabelle 3.5: Die Kommandozeilenparameter von fdisk

Im bisherigen Abschnitt war von »Partitionen« die Rede. Ab sofort wird dieser Begriff in einem anderen Zusammenhang verwendet, um die Verwirrung etwas zu steigern. Es

handelt sich in der BSD-Begriffswelt nämlich um »Windows-Partitionen«, auch wenn Sie aus Zeit stammen, wo noch niemand an Windows gedacht hat und auch Linux mit dieser Art von Festplattenaufteilung arbeitet.

BSD, Solaris und auch SCO Unixware verwenden den Begriff »Partition« aber anders. Diese Windows-Partitionen heißen bei diesen Betriebssystemen »Slices«, bei NetBSD findet außerdem noch der synonyme Begriff »Partition Slot« Verwendung. Außerdem ist darauf zu achten, daß die Partitionstabelle als »Partition Table« bezeichnet wird. Die Begriffe werden also etwas verschwommen eingesetzt.

Bei der Aufteilung der Festplatte unter NetBSD wird immer eine primäre Partition (besser formuliert »ein primärer Slice«) verwendet. NetBSD kann nicht in eine erweiterte »Windows-Partition« installiert werden. Im Gegensatz zu Linux beispielsweise, das sich auch in erweiterten Laufwerken installieren läßt, nehmen Net-, Free- und OpenBSD immer genau einen primären Slice pro Festplatte, der dann intern weiter aufgeteilt wird. Um die Verwirrung auf die Spitze zu treiben: Diese weiteren Aufteilungen werden unter BSD als Partitionen bezeichnet.

Eine Festplatte wird unter NetBSD immer folgendermaßen angesprochen: *Schnittstelle-Plattennummer-Partitionsnummer*, umgesetzt in die echte Buchstabenkombination wäre das dann beispielsweise *sd0a* oder *wd1f*. Ersteres bedeutet SCSCI-Device *0*, Partition *a* und das zweite Gerät ist IDE-Device *1*, Partition *f*. Es ist zu beachten, daß die Slice-ID im Gegensatz zur Kennung bei FreeBSD unterschlagen wird. Zählungen der Festplatte (und auch der Slices) beginnen C-gemäß immer bei Null.

Die Unterdrückung der Slice-Kennung bedeutet aber auch, daß (wieder im Gegensatz zu FreeBSD) Fremdpartitionen, damit sie mit dem *mount*-Befehl eingebunden werden können, in den Verwaltungsbereich des Betriebssystems, das Disklabel, aufgenommen werden müssen. Dieses Disklabel ist ein elementarer Teil der Festplattenstrukturierung bei BSD.

3.3.1 Festplattenaufteilung mit fdisk

Für die Grobeinteilung der Festplatte in Slices ist in NetBSD das Programm *fdisk* zuständig. Im Gegensatz zur recht komfortablen Plattenaufteilung im NetBSD-Installationsprogramm ist das reguläre *fdisk* aber sehr spartanisch und gewöhnungsbedürftig. Es kennt die folgenden Aufrufparameter von Tabelle 3.5.

Im Gegensatz zu den meisten *fdisk*-Programmen anderer Betriebssysteme läßt sich *fdisk* unter NetBSD auch ganz ohne Parameter aufrufen. Es zeigt dann die Partitionierungsdaten der aktuellen Bootfestplatte an:

```
[~]#fdisk
Disk: /dev/rwd1d
NetBSD disklabel disk geometry:
cylinders: 10672 heads: 15 sectors/track: 63 (945 sectors/cylinder)

BIOS disk geometry:
cylinders: 1023 heads: 15 sectors/track: 63 (945 sectors/cylinder)
```

```
Partition table:
0: sysid 169 (NetBSD)
    start 63, size 10084914 (4924 MB), flag 0x80
        beg: cylinder    0, head   1, sector  1
        end: cylinder  431, head  13, sector 63
1: <UNUSED>
2: <UNUSED>
3: <UNUSED>
```

Will man die Daten einer anderen Festplatte lesen, muß deren Name angegeben werden:

```
[~]#fdisk wd2
Disk: /dev/rwd2d
NetBSD disklabel disk geometry:
cylinders: 16383 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

BIOS disk geometry:
cylinders: 1023 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

Partition table:
0: sysid 169 (NetBSD)
    start 63, size 10233342 (4996 MB), flag 0x80
        beg: cylinder    0, head   1, sector  1
        end: cylinder  636, head 254, sector 63
1: sysid 11 (Primary DOS with 32 bit FAT)
    start 10233405, size 6217155 (3035 MB), flag 0x0
        beg: cylinder  637, head   0, sector  1
        end: cylinder 1022, head 254, sector 63
2: <UNUSED>
3: <UNUSED>
```

Oder so:

```
[~]#fdisk wd0
Disk: /dev/rwd0d
NetBSD disklabel disk geometry:
cylinders: 13410 heads: 15 sectors/track: 63 (945 sectors/cylinder)
BIOS disk geometry:
cylinders: 1023 heads: 15 sectors/track: 63 (945 sectors/cylinder)

Partition table:
0: sysid 169 (NetBSD)
    start 63, size 12672324 (6187 MB), flag 0x0
        beg: cylinder    0, head   1, sector  1
        end: cylinder   97, head  13, sector 63
```

```
1: <UNUSED>
2: <UNUSED>
3: <UNUSED>
```

An der Angaben ist zu sehen, daß das System mehrere IDE-Festplatten eingebaut hat. Jede der Platten besitzt eine NetBSD-Partition, nur die dritte Platte zusätzlich noch eine DOS-Partition. Gebootet wird von der zweiten Platte (»wd1«). Dem System stehen über die drei Festplatten verteilt zirka 15 Gbyte Speicher zur Verfügung.

Will man das ändern, ruft man *fdisk* im interaktiven Modus (*-u*) auf:

```
[~]#fdisk -u wd2
Disk: /dev/rwd2d
NetBSD disklabel disk geometry:
cylinders: 16383 heads: 16 sectors/track: 63 (1008 sectors/cylinder)
BIOS disk geometry:
cylinders: 1023 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

Do you want to change our idea of what BIOS thinks? [n]
Partition table:
The data for partition 0 is:
sysid 169 (NetBSD)
    start 63, size 10233342 (4996 MB), flag 0x80
        beg: cylinder    0, head   1, sector  1
        end: cylinder  636, head 254, sector 63
Do you want to change it? [n]
The data for partition 1 is:
sysid 11 (Primary DOS with 32 bit FAT)
    start 10233405, size 6217155 (3035 MB), flag 0x0
        beg: cylinder  637, head   0, sector  1
        end: cylinder 1022, head 254, sector 63
Do you want to change it? [n] y
sysid: [11]
```

Hier ist es jetzt möglich, ohne weitere Änderung der Größe der Partition eine andere ID zu geben, zum Beispiel die Nummer 131 für Linux ext2fs:

```
sysid: [11] 131
start: [10233405]
size: [6217155]
Explicitly specify beg/end address? [n]
sysid 131 (Linux native)
    start 10233405, size 6217155 (3035 MB), flag 0x0
        beg: cylinder 1022, head  15, sector 63
        end: cylinder 1022, head  15, sector 63
Is this entry okay? [n] y
```

Auch mit dieser Eintragung ist noch nichts verloren:

```
The data for partition 2 is:
<UNUSED>
Do you want to change it? [n]
The data for partition 3 is:
<UNUSED>
Do you want to change it? [n]

We haven't written the MBR back to disk yet.  This is your last chance.
Disk: /dev/rwd2d
NetBSD disklabel disk geometry:
cylinders: 16383 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

BIOS disk geometry:
cylinders: 1023 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

Partition table:
0: sysid 169 (NetBSD)
    start 63, size 10233342 (4996 MB), flag 0x80
        beg: cylinder    0, head   1, sector  1
        end: cylinder  636, head 254, sector 63
1: sysid 131 (Linux native)
    start 10233405, size 6217155 (3035 MB), flag 0x0
        beg: cylinder 1022, head  15, sector 63
        end: cylinder 1022, head  15, sector 63
2: <UNUSED>
3: <UNUSED>
Should we write new partition table? [n]
```

Erst wenn hier mit [y] bestätigt wird, schreibt *fdisk* die Daten auf die Festplatte. Geprüft wird das, wenn man *fdisk* nach dem Abschluß noch einmal ohne den Parameter *-u* aufruft:

```
[~]#fdisk wd2
Disk: /dev/rwd2d
NetBSD disklabel disk geometry:
cylinders: 16383 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

BIOS disk geometry:
cylinders: 1023 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

Partition table:
0: sysid 169 (NetBSD)
    start 63, size 10233342 (4996 MB), flag 0x80
        beg: cylinder    0, head   1, sector  1
```

```
        end: cylinder  636, head 254, sector 63
1: sysid 131 (Linux native)
    start 10233405, size 6217155 (3035 MB), flag 0x0
        beg: cylinder 1022, head  15, sector 63
        end: cylinder 1022, head  15, sector 63
2: <UNUSED>
3: <UNUSED>
```

Bedenken Sie, daß das Editieren einer vorhandenen Partition – egal, ob Slice oder BSD-Partition – riskant ist und mit einem Datenverlust enden kann. Nehmen Sie also immer *vorher* das längst überfällige Backup der Daten vor!

Was in diesem Fall auch stutzig machen sollte, ist, daß für die neue Partition als Start- und Enddaten die selben Werte eingetragen sind. Das ist bei Fremddateisystemen aber kein Problem.

Warum wurde aber diese DOS-Partition umgewandelt? Der Grund ist einfach: Das MS-DOS-Dateisystem (FAT oder VFAT) ist für den Datenaustausch nicht unpraktisch. Es macht aber keinen Sinn, auf einer dritten Festplatte eine DOS-Partition zu halten, weil DOS/Windows niemals darauf bootfähig sein wird. Aber FAT hat noch eine andere Schwäche: es kommt weder mit Unix-Links noch mit Unix-Dateiattributen zurecht. Linux-Partitionen erfüllen diese Bedingungen und können unter NetBSD problemlos verwendet werden. Wie sie formatiert und gewartet werden, wird noch beschrieben.

3.3.2 Das BSD-Disklabel

Im letzten Abschnitt war davon die Rede, daß bei BSD die Aufteilung über Slices und BSD-Partition erfolgt. Wenn man NetBSD installiert, wird man von *sysinstall* auch gefragt, wie man das Disklabel benennen will und daß man die Voreinstellung *mylabel* durchaus beibehalten kann.

Der Name des Disklabels ist in der Tat bedeutungslos. Das Disklabel selbst ist für NetBSD aber elementar. BSD-Partitionen werden über Buchstabenkennungen (*a*, *e*, *f* und so weiter) gemountet. Das sind die Einträge in diesem Disklabel. Die Kennungen *a* bis *d* sind fest vorgegeben, die restlichen dürfen frei vergeben werden.

Im Disklabel von NetBSD (und OpenBSD) finden im Gegensatz zu FreeBSD auch Fremdpartitionen Platz. Das stellte bis zur Version 1.5 bei NetBSD ein Problem dar, da das Disklabel nur maximal acht Einträge besitzen durfte; seit Version 1.6 wurde die Zahl der möglichen Einträge auf sechzehn erhöht (wie schon früher bei OpenBSD). Platzprobleme im Disklabel dürften damit der Vergangenheit angehören.

Auch dadurch, daß Fremdpartitionen in das Disklabel aufgenommen werden müssen und solche ja durchaus nach der Installation von NetBSD angelegt werden könnten, kommt dem Disklabel-Editor eine zentrale Bedeutung bei der Festplattenverwaltung zu. Auch für Umwandlung von FFS-Partitionen in das LFS-Dateisystem wird er benötigt. Der Disklabel-Editor besitzt den Namen *disklabel* und kennt zahlreiche Parameter. Sie sollten hier nicht alle aufgelistet werden. Die seltener eingesetzten sollten Sie in der Manpage nachlesen.

Label-Kennung	Funktion
a	Bootpartition (»/«)
b	Speicherauslagerungsbereich (Swap)
c	Der gesamte Slice
d	Die komplette Platte

Tabelle 3.6: Fest zugewiesene und nicht änderbare Partitionskennungen bei NetBSD/i386

```
    IW   /tmp//EdDk.a01767a          Row 1    Col 1   10:53  Ctrl-K H for help
# /dev/rwd2d:
type: unknown
disk: mydisk3
label:
flags:
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 26588016
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 b:   394065        63       swap                    # (Cyl.    0*- 390)
 c: 10233342        63     unused      0     0       # (Cyl.    0*- 10152*)
 d: 26588016         0     unused      0     0       # (Cyl.    0 - 26376)
 e:  9839277    394128     4.2BSD   1024  8192    86  # (Cyl.  391 - 10152*)
 h:  6217155  10233405     unknown                    # (Cyl. 10152*- 16319)

** Joe's Own Editor v2.8 ** Copyright (C) 1995 Joseph H. Allen **
```

Abb. 3.18: Ein NetBSD-1.6-Disklabel im Editor (i386-Port)

Der normale (interaktive) Modus wird durch den Aufruf mit *disklabel -e Plattenkennung* gestartet. Wie oft bei NetBSD werden dann die Daten in den Systemeditor geladen und dort bearbeitet. Speichert man und verläßt man den Editor, werden diese modifizierten

Daten zurückgeschrieben. Beim Zurückschreiben wird auch geprüft, ob die neuen Angaben sinnvoll sind und gegebenenfalls um einen neuen Editorlauf gebeten.

Aus den Angaben in Abbildung 3.18 ist zu entnehmen, daß diese Festplatte keine NetBSD-Bootpartition enthält (*a* fehlt) und außer einer Partition im FFS-Format noch eine Partition *h* unbekannten Typs enthält.

Es ist in diesem Zusammenhang wichtig zu wissen, daß *disklabel* von NetBSD vor der Version 1.6 nur vier Partitionstypen kennt: *4.2BSD*, was FFS entspricht, *4.4LFS*, was für das LFS-Dateisystem steht, *swap* für Auslagerungspartitionen und *unknown*. *unknown* bedeutet in diesem Zusammenhang nicht, daß diese Partition nicht eingebunden werden kann. Der Typ ist nur dem Disklabel-Editor nicht bekannt. Welche Partitionstypen in der aktuellen NetBSD-Version bekannt sind, erfahren Sie, wenn Sie *disklabel* mit dem Parameter *-i* aufrufen und dann in diesem Modus nach der Auswahl einer Partition ein Fragezeichen angeben:

```
#disklabel -i /dev/wd0
partition> a
Filesystem type [?] [4.2BSD]: ?
Supported file system types:
        4.1BSD          Eighth Edition  MSDOS           unused
        4.2BSD          FILECORE        NTFS            Version 6
        4.4LFS          HFS             RAID            Version 7
        ADOS            HPFS            swap
        boot            ISO9660         System V
        ccd             Linux Ext2      unknown
```

Um die FFS-Partition *e* in eine LFS-Partition zu konvertieren, muß nur der Eintrag der Definition geändert werden in

```
  e:   9839277   394128   4.4LFS   1024  8192   43   # (Cyl.  391 - 10152*)
```

Der hohe Wert bei *cpg/sgs* zeigt übrigens an, daß diese Platte unter NetBSD 1.6 formatiert wurde. Bei einer unter NetBSD 1.5.2 formatierten Platte (Abb. 3.19) werden die (alten) Standardwerte angezeigt.

Insgesamt spielt es aber keine Rolle, mit welcher Version von NetBSD die Festplatte formatiert wurde. NetBSD 1.5.x kommt mit dem neuen Disklabel sehr gut zurecht, wenn es maximal acht Einträge besitzt.

Im Gegensatz zum Programm *fdisk*, das es auf manchen Plattformen wie beispielsweise NetBSD/mac68k nicht gibt, ist *disklabel* immer im Lieferumfang enthalten. Abbildung 3.20 zeigt die Ausgabe des Programms auf einem Macintosh.

```
    IW   /tmp//EdDk.a01801a          Row 1    Col 1   11:08  Ctrl-K H for help
# /dev/rwd0d:
type: unknown
disk: mydisk
label:
flags:
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 15
sectors/cylinder: 945
cylinders: 13410
total sectors: 12672450
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a:    436527        63     4.2BSD   1024  8192    16   # (Cyl.    0*- 461)
 b:    788130    436590       swap                       # (Cyl.  462 - 1295)
 c:  12672324        63     unused      0     0         # (Cyl.    0*- 13409*)
 d:  12672450         0     unused      0     0         # (Cyl.    0 - 13409)
 e:  11447667   1224720     4.2BSD   1024  8192    16   # (Cyl. 1296 - 13409*)

** Joe's Own Editor v2.8 ** Copyright (C) 1995 Joseph H. Allen **
```

Abb. 3.19: Ein Disklabel von NetBSD 1.5.2 im Editor von NetBSD 1.6 (i386-Port)

Anders als bei der Intel-Plattform steht beim Macintosh-Port der Buchstabe *d* nicht für die gesamte Festplatte, sondern wird auch für die Eintragung einer Partition verwendet[1]. Haben Sie die Änderungen an der Festplattenaufteilung vorgenommen oder die neue Festplatte eingerichtet, müssen die neuen Partitionen formatiert werden.

Je nach Dateisystem finden unter NetBSD unterschiedliche Formatierprogramme Einsatz. Ihnen allen gemeinsam ist, daß als Parameter die Kennung im Disklabel angegeben werden muß. Es können also keine Festplatten-Slices formatiert und eingebunden werden, die nicht darin eingetragen sind.

[1] Es hilft im Fall des Macintosh aber nichts, daß die HFS-Partition ins Disklabel eingefügt ist, da Apple-HFS-Partitionen nicht gemountet werden können.

```
[~] #disklabel sd0
# /dev/rsd0c:
type: SCSI
disk: DCAS-34330
label: fictitious
flags:
bytes/sector: 512
sectors/track: 171
tracks/cylinder: 6
sectors/cylinder: 1026
cylinders: 8205
total sectors: 8467200
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

8 partitions:
#        size   offset     fstype  [fsize bsize cpg/sgs]
 a:    423360       96     4.2BSD       0     0     0   # (Cyl.    0*- 412*)
 b:    423360  5249704       swap                       # (Cyl. 5116*- 5529*)
 c:   8467200        0     unused       0     0         # (Cyl.    0 - 8252*)
 d:   3810196   423456        HFS                       # (Cyl.  412*- 4126*)
 e:   1058386  5673064     4.2BSD       0     0     0   # (Cyl. 5529*- 6560*)
 f:   1016052  6731450     4.2BSD       0     0     0   # (Cyl. 6560*- 7551*)
 g:   1016052  4233652     4.2BSD       0     0     0   # (Cyl. 4126*- 5116*)
 h:    719698  7747502        HFS                       # (Cyl. 7551*- 8252*)
disklabel: boot block size 0
disklabel: super block size 0
```

Abb. 3.20: Der Befehl disklabel bei NetBSD/mac68k 1.6

Programm	Dateisystem
newfs	Formatiert eine native BSD-Partition im FFS-Format.
news_lfs	Formatiert eine Partition im Log structured Dateisystem.
newfs_msdos	Formatiert eine DOS-FAT/VFAT-Partition.
mkfs.ext2fs	Formatiert eine Linux-Ext2-Partition. Das Programm wird über das Package »e2fsprogs« installiert.

Tabelle 3.7: Die verschiedenen Formatierprogramme in NetBSD

Im Gegensatz zu FreeBSD 4 ist das Prüfprogramm für Linux-Partitionen bei NetBSD im Grundsystem enthalten. Äquivalent zum Formatierprogramm stehen auch die Reinigungsprogramme zur Verfügung. Allerdings gibt es für Linux-Dateisysteme nach der sehr zu empfehlenden Installation des Packages »e2fsprogs« gleich drei Stück, darunter eines für das ext3-Dateisystem, das unter NetBSD selbst nicht angelegt, sondern nur genutzt werden kann.

Wurden alle Partitionen nach Wunsch angelegt und erfolgreich getestet, sollten sie auch dem System bekannt gegeben werden. Dazu werden sie in die Datei »/etc/fstab« eingetragen. Im Gegensatz zu Linux, in dem mehrere Dateisystemtabellen gehalten werden, gibt es unter NetBSD nur eine. Sie wird beim Systemstart eingelesen und ausgewertet. Deshalb müssen die Einträge unbedingt stimmen.

Programm	Funktion
fsck_ffs	Reinigungsprogramm für FFS-Datenträger.
fsck_lfs	Reinigungsprogramm für LFS-Datenträger.
fsck_msdos	Reinigungsprogramm für FAT-Datenträger.
fsck_ext2fs/fsck.ext2	Reinigungsprogramme für Linux-Ext2-Datenträger. *fsck.ext2* wird über das Package »e2fsprogs« installiert.
fsck.ext3	Reinigungsprogramm für Linux-Ext3-Datenträger. Das Programm wird über das Package »e2fsprogs« installiert.

Tabelle 3.8: Die bei NetBSD mitgelieferten Dateisystem-Reinigungsprogramme

Die folgende »fstab« stammt von einem PC und zwar von der Maschine, bei der weiter oben mit *fdisk* eine DOS- in eine Linux-Partition umgewandelt wurde:

```
# Device    mnt-point     fstype    size & options           checks
# ------------------------------------------------------------------
/dev/wd1a   /             ffs       rw                       1 1
/dev/wd1b   none          swap      sw                       0 0
/dev/wd0b   none          swap      sw                       0 0
/dev/wd2b   none          swap      sw                       0 0
/dev/wd1e   /usr          ffs       rw                       1 2
/dev/wd2e   /pub          ffs       rw,userquota,groupquota  1 1
kernfs      /kern         kernfs    rw                       0 0
procfs      /proc         procfs    rw                       0 0
/dev/cd0d   /mnt/cdrom    cd9660    ro,noauto                0 0
/dev/wd2h   /mnt/linux    ext2fs    rw,noauto                0 0
/dev/fd0a   /mnt/floppy   ffs       rw,noauto                0 0
/dev/fd0a   /mnt/A:       msdos     rw,noauto                0 0
```

Die erste Platte, die ein NetBSD 1.5.2 enthält, ist nicht in die Datei eingetragen und wird damit auch nicht eingebunden.

Die folgende Datei stammt von einem Macintosh unter NetBSD/mac68k 1.6:

```
# Device    mnt-point     fstype    size & options          checks
# ------------------------------------------------------------------
/dev/sd0a   /             ffs       rw                      1 1
/dev/sd0b   none          swap      sw                      0 0
/dev/sd0g   /usr/X11R6    ffs       rw                      1 1
/dev/sd0f   /usr/pkgsrc   ffs       rw                      1 1
/dev/sd0e   /usr/pkg      ffs       rw                      1 1
/dev/cd0d   /mnt/cdrom    cd9660    ro,noauto               0 0
proc        /proc         procfs    rw                      0 0
kern        /kern         kernfs    rw                      0 0
```

Die dritte »fstab«-Datei hält sich nicht mehr an irgendwelche Standards. Sie stammt von der CD-ROM-Distribution der Buch-CD:

```
# Device   mnt-point   fstype   size & options                  checks
# ------------------------------------------------------------------
swap       /dev        mfs      rw,union,-s=65536                0 0
swap       /tmp        mfs      rw,nodev,noexec,nosuid,-s=131072 0 0
swap       /var        mfs      rw,-s=131072,noatime             0 0
swap       /etc        mfs      rw,-s=65536                      0 0
swap       /home       mfs      rw,-s=131072                     0 0
swap       /root       mfs      rw,-s=65536                      0 0
kernfs     /kern       kernfs   rw                               0 0
procfs     /proc       procfs   rw                               0 0
/dev/fd0a  /mnt/floppy1 msdos   rw,noauto                        0 0
```

Nach so vielen Dateiabbildungen folgt nun auch die Erläuterung:

Eine Zeile in der Datei enthält zuerst immer das einzubindende Gerät in »/dev«, dann den Mountpunkt, also das Verzeichnis, in das das Gerät eingehängt wird. Anschließend sind das Dateisystem und die Mountoptionen angegeben. Zum Schluß wird festgelegt, wie Dateisystem-Dumps und die Dateisystemprüfung abzulaufen haben.

Eine Ausnahme von der Regel, daß das Device angegeben wird, stellen RAM-Disks (*mfs*) sowie die Pseudo-Dateisysteme *kernfs* und *procfs* dar. Hier wird der Typ (*swap*, *kernfs* und *procfs*) angegeben. Es gibt ja kein Gerät, von dem Daten eingebunden werden. Die Dateisysteme werden leer beim Start erzeugt. Hier wird aber angegeben, wo die Pseudogeräte eingebunden werden.

Genau umgekehrt funktioniert es bei Auslagerungsbereichen. Diese werden selbstverständlich nicht gemountet. Deshalb steht beim Mountpunkt die Kennung *none*.

Die Verwaltung der Auslagerungspartitionen wird durch die Programme *swapctl* und *swapon* gesteuert. *swapon -a* bindet alle in der Datei »/etc/fstab« angegebenen Swapbereiche ein. *swapon* ist nur ein Subset von *swapctl*, die Verwaltung läßt sich deshalb auch komplett mit dem allgemeinen Programm durchführen. Seine wichtigsten Parameter sind:

-A	Bindet alle in der »fstab« angegebenen Swapbereiche ein.
-U	Entfernt alle in der »fstab« angegebenen Swapbereiche.
-l	Listet alle momentan eingebundenen Swapbereiche auf.
-s	Anzeige einer einzeiligen Swapstatistik.

Die weiteren zahlreichen Parameter des Programms erfahren Sie aus der Manpage.

Haben Sie keinen Swap-Bereich definiert oder eine zu kleine Auslagerungspartition eingerichtet, muß deshalb nicht die Festplatte neu partitioniert werden. NetBSD erlaubt auch die Einrichtung von Swapdateien. Auch sie müssen in der Datei »fstab« angegeben werden, beispielsweise so:

```
/var/swapfile   none        swap     sw,priority=1              0 0
```

Das System reagiert sehr ungehalten auf mangelnden Speicher. Sollte er bei großer Belastung ausgehen, ist die Definition einer Swap-Datei ein probates Mittel, um den Abbruch von Programmen wegen Speichermangels zu verhindern. Der Swapbereich kann auch ohne Neustart eingebunden werden. Dazu wird nur die Swap-Zeile eingefügt und anschließend *swapon -a* oder *swapctl -A* aufgerufen. Der Parameter *-A* von *swapctl* bindet nur Swap-Partitionen, der Parameter *-a* zusätzlich auch Swap-Dateien ein. *swapon -a* ist identisch zum Start von *swapctl -A*.

Bevor eine Auslagerungsdatei verwendet werden kann, muß sie aber angelegt werden[1]. Außerdem muß sie die Attribute 0600 haben. Eine FAT-Partition scheidet deshalb für das Anlegen im Normalfall aus. Die folgenden Zeilen bereiten eine zehn Mbyte große Swapdatei vor:

```
dd if=/dev/zero bs=1m of=/var/swapfile count=10
chmod 0600 /var/swapfile
```

[1] Man kann mit so einer generierten Datei erheblich mehr machen als nur eine Swapdatei zu erzeugen. Es ist sogar möglich, eine solche Datei mit *fdisk*, *disklabel* und *newfs* zu bearbeiten und damit eine virtuelle Festplatte zu erzeugen die sich auch auf eine CD brennen läßt. Wenn Sie sich für diese Speziallösung interessieren, sollten Sie den Beitrag dazu in der Ausgabe 2/2003 der freeX nachlesen.

Ein Aufruf von *swapctrl -l* listet nach dem Aufruf von *swapon -a* alle verwendeten Auslagerungsbereiche auf:

```
[~]#swapctl -l
Device              1K-blocks     Used    Avail Capacity  Priority
/dev/wd1b              394065    14944   379121     4%    0
/dev/wd0b              394065    15656   378409     4%    0
/dev/wd2b              197032    15784   181248     8%    0
/var/swapfile           10240        0    10240     0%    0
Total                  995402    46384   949018     5%    1
```

Ein kleines Shellskript, das die Arbeiten automatisiert, könnte so aussehen:

```
#!/bin/sh
if [ "$1" = "" ] ; then
  echo "No position for swap file specified. Exit."
  exit 1
fi

if [ "$2" = "" ] ; then
  COUNT=10            # Default: 10 Mbyte
else
  COUNT=$2
fi

echo -n "Creating ${COUNT} m swapfile $1/swapfile ... "
dd if=/dev/zero bs=1m of=$1/swapfile count=$COUNT > /dev/null 2>&1
echo "done."

chmod 0600 /$1/swapfile

echo -n "Adding $1/swapfile to swaplist ..."
swapctl -p 1 -a $1/swapfile
echo "done."
swapctl -l
```

Entfernt wird ein Swapbereich mit dem Parameter *-d* von *swapctl*:

```
swapctl -d /var/swapfile
```

Die »priority« braucht übrigens bei der Swapdatei nicht unbedingt angegeben zu werden. Läßt man den Parameter in der Datei »fstab« weg, taucht er bei *swapctl -l* auch nicht auf.

mount_union und mount -t union

Alle BSD-Betriebssysteme besitzen die Dateisystemangabe *union*. Dabei handelt es sich nicht um ein echtes Dateisystem, das an beliebiger Stelle eingebunden wird. Vielmehr werden hiermit zwei verschiedene gemountete Verzeichnisse zusammengeführt. Unions sind äußerst instabil und sollten nur mit äußerster Vorsicht eingesetzt werden. Sie führen im Extremfall zum Totalabsturz des Betriebssystems. Sie sind in anderer Hinsicht aber ein elegantes Mittel, einen primär schreibgeschützten Datenträger beschreiben zu können. Das zeigt folgendes Szenario:

Die Daten von einem CD-Dateisystem sollen um weitere Daten ergänzt werden. Natürlich kann auf die CD nicht geschrieben werden. Deshalb wird ein normales Verzeichnis über die CD-ROM gemountet:

```
# mkdir /pub/newdate
# mount_cd9660 /dev/cd0d /pub/cddata
# mount_union /pub/newdata /pub/cddata
```

Diese Vorgehensweise ist beispielsweise dann praktisch, wenn die CD einen auf einem anderen Rechner erstellten CVS-Checkout enthält, den man lokal aktualisieren will. Nach dem erneuten Checkout befinden sich nur die modifizierten Daten lokal auf der Festplatte. Mit den kompletten Daten kann aber, obwohl eine CD-ROM unterliegt, normal gearbeitet werden.

Union-Mounts müssen unbedingt vor dem Shutdown mit *umount* manuell wieder aufgelöst werden. Sie sollten außerdem nicht automatisch über die »/etc/fstab« erfolgen und bei Nicht-Bedarf auch möglichst schnell wieder entfernt werden.

Die anderen Spezialdateisysteme wie *nullfs* funktionieren ähnlich. Konsultieren Sie bei Bedarf die entsprechenden Manpages.

Der Dateisystembereich

Zu Device und Mountpunkt muß jeweils angegeben sein, welches Dateisystem der einzubindende Datenträger besitzt. Als Angabe können alle Dateisysteme dienen, die einen entsprechenden Mount-Befehl in NetBSD besitzen: *ffs/ufs*, *lfs*, *msdos* (FAT/VFAT), *cd9660*, *ntfs*, *ext2* (inklusive *ext3*), *mfs* und *ados*. Die Dateisysteme *union*, *null*, *portal*, *filecore*, *overlay* und *umap* spielen in diesem Zusammenhang keine Rolle. Es sind spezielle zum Teil experimentelle Lösungen. *procfs* und *kernfs* sind Spezialdateisysteme zu den Prozessen und des Kernelstatus. *procfs* ist vorwiegend für Linux-Programme wichtig, *kernfs* für die Kompatibilität zu FreeBSD-Programmen.

CD-ROMs und User-Mounts

NetBSD erlaubt in der Voreinstellung nur das Mounten von Dateisystemen durch den Systemadministrator *root*. Bei Festplattenpartitionen ist das sinnvoll. Bei CD-ROMs ist das aber in der Regel unpraktisch. Es ist aber mit wenigen Handgriffen möglich, dies zu ändern. Dazu muß im ersten Schritt durch *root* Gruppen und anderen der Schreib-/Lesezugriff auf das CD-Device gegeben werden:

```
# chmod go+rw /dev/cd0d
```

Im zweiten Schritt muß den Usern die Erlaubnis erteilt werden, Verzeichnisse einzubinden:

```
# sysctl -w vfs.generic.usermount=1
vfs.generic.usermount: 0 -> 1
```

Im letzten Schritt muß der User den Mountpunkt besitzen. Dazu muß er in seinem lokalen Verzeichnisbereich den Mountpunkt anlegen, beispielsweise mit

```
$ cd
$ mkdir cdrom
```

Hier kann dann die CD-ROM eingehängt werden:

```
$ /sbin/mount_cd9660 /dev/cd0d ~/cdrom
```

Die sysctl-Angabe ist nur bis zum nächsten Neustart gültig. Sie muß deshalb in die Datei »/etc/sysctl.conf« eingetragen werden:

```
vfs.generic.usermount=1
```

Der Optionenbereich

Entscheidende Rolle beim Verarbeiten der Datei »fstab« hat der Optionenbereich. Hier muß grundsätzlich angegeben werden, ob die Partition schreibbar ist. Es stehen die Parameter in Tabelle 3.9 zur Verfügung.

Eigene Partitionen im FFS- und LFS-Dateisystem werden im Normalfall für das Lesen und Schreiben eingebunden. Bei Fremddateisystemen wird das normalerweise nicht empfohlen. Solange ein Prüfprogramm (*msdos*, *ext2*, *ext3*) zur Verfügung steht, ist es aber problemlos möglich. Es können aber leicht Schwierigkeiten auftreten, wenn beispielsweise durch einen Stromausfall während des Betriebs von NetBSD eine so geöffnete Partition in einem nicht-konsistenten Zustand belassen wird. Sie kann anschließend nicht von NetBSD aus repariert werden. So findet sich in der Man-Page von *mount_ntfs* beispielsweise der folgende Hinweis: *»Warning: do not mount NTFS filesystems read-write.*

The write support is not very useful and is not tested well. It's not safe to write to any file on NTFS, you might damage the filesystem. Unless you want to debug NTFS filesystem code, mount the NTFS filesystem read-only.«

Kürzel	Bedeutung	Beschreibung
rw	read/write	Schreib- und Leseberechtigung werden erteilt.
ro	read only	Es ist nur das Lesen erlaubt. Keine Schreibberechtigung. Sinnvoll ist das bei Fremddateisystemen und natürlich bei CD-ROMs.
sw	swap	Kennung für einen Auslagerungsbereich (Swap-Partition oder Swap-Datei).
dp	dump	Kennung für Dump Devices.
xx	ignore	Laufwerk ignorieren, es ist unbenutzt.

Tabelle 3.9: Pflichtoptionen zur Behandlung der Partititon in der »/etc/fstab«

Weitere Optionen werden hinter der Schreib-/Lese-Definition durch Kommata getrennt angegeben. Tabelle 3.10 zeigt einige Beispiele.

Dump und Check

Die letzte Angabe in den Zeilen der Datei »fstab« sind zwei Zahlen. Die erste definiert, ob das Dateisystem vom Backupbefehl *dump* zu berücksichtigen ist. Falls hier eine 0 steht, wird das Dateisystem ignoriert.

Die zweite Zahl wird vom Dateisystem-Prüfprogramm *fsck* ausgewertet. Angegeben wird der Lauf, bei dem der Datenträger bearbeitet wird, 0 bedeutet, daß das Dateisystem nicht geprüft wird. Normalerweise werden das Root-Dateisystem im ersten Durchlauf (1) und die anderen Dateisysteme im zweiten Lauf (2) geprüft. Befinden sich die Laufwerke auf einer physikalischen Festplatte, erfolgen die Prüfungen nacheinander, befinden sie sich auf unterschiedlichen Datenträgern, werden die Prüfungen parallel durchgeführt.

3.3.3 Das Spezialverzeichnis »/dev«

Im Gegensatz zu Linux 2.4 und FreeBSD 5 besitzt NetBSD keine dynamische Geräteverwaltung. Das bedeutet, daß nicht der Kernel beim Booten die Devices anlegt, sie sind wie bei Linux 2.2 und FreeBSD bis Version 4.x statisch. Prinzipiell sind im Verzeichnis »/dev« also auch Einträge vorhanden, die einem nichtkonfigurierten Gerät entsprechen. Da es kein »devfs«-Dateisystem gibt, muß gewährleistet sein, daß neue Geräte auch hier vorhanden sind. Dafür enthält das Verzeichnis ein Skript *MAKEDEV*, das mit verschiedenen Parametern aufgerufen einen Teil der Gerätedateien anlegt. Mit

```
#cd /dev
#./MAKEDEV all
```

als User *root* ist man auf der sicheren Seite. Es werden alle möglichen Device-Knoten erzeugt. Andere Parameter sind in der Skriptdatei selbst erläutert. So erzeugt *./MAKEDEV local* zum Beispiel alle lokalen Knoten.

Option	Beschreibung
noauto	Der Datenträger wird beim Booten und bei der Angaben von *mount -a* (Einbinden aller Laufwerke) nicht mitbehandelt. So sollten alle Wechseldatenträger (Disketten, CD-Laufwerke, Wechselplatten) definiert sein.
userquota, groupquota	Auf dem Laufwerk werden User- und Gruppenquotas verwendet. Diese Parameter benötigen Quota-Unterstützung im Kernel und entsprechende Quota-Dateien. Befinden sich die Quota-Dateien an ein einem anderen Platz als dem Hauptverzeichnis des Laufwerks oder sollen sie einen anderen als den voreingestellten Namen erhalten, ist dieser mit einem Gleichheitszeichen hinter der jeweiligen Definition mit dem vollen Pfadnamen anzugeben.
union	Das Laufwerk wird union-gemountet. Damit bleiben vorhandene Dateien im übergemounteten Verzeichnis erhalten. Union-Mounts sind nicht unbedingt sicher. Union-Mounts auf das Hauptverzeichnis führen auch mit großer Wahrscheinlichkeit zum Absturz des Betriebssystems.
-s=65536	Die Größe s= gibt bei RAM-Disks die maximale (nicht die initiale) Grösse des Pseudo-Datenträgers in Kbyte an. Hier sind es also 64 Mbyte. Die Größe kann genauso auch in Mbyte angegeben werden. Dann muß nur als Kennung nach der Größenangabe ein »m« angegeben sein, beispielsweise *-s=64m*.
nodev, noexec, nosuid, noatime	*nodev* = Gerätedateien auf dem Device werden nicht interpretiert, *noexec* = Binärdateien werden nicht ausgeführt, *nosuid* = Das Setzen von GUID- und SUID-Bits ist nicht möglich, *noatime* = Das Zugriffsfeld (letzter Dateizugriff) wird nicht aktualisiert. Es sind weitere Angaben möglich, die Sie der Manpage von *mount* entnehmen können. Im Normalfall werden sie nicht gebraucht.

Tabelle 3.10: Einige ausgewählte mögliche Parameter der fstab-Optionen

Das Verzeichnis »/dev« muß sich (wie »/etc«, »/bin« und »/sbin«) immer auf der Rootpartition befinden, da sonst die notwendigen Devices nicht verfügbar sind und das System nicht starten kann. Es kann auch nicht beim Mounten ersetzt werden. Ein Teil der Geräte in »/dev« muß außerdem beschrieben werden können. Es ist also nicht möglich, dieses Verzeichnis auf einem schreibgeschützten Datenträger zu halten. Wenn man einen schreibgeschützten Datenträger mounten muß, hilft es aber, wie in der fstab-

Datei von NetBSD-Live weiter oben gezeigt, einen Union-MFS-Mount durchzuführen. Damit werden die neuen Einträge über die ursprünglichen gemappt.

Union-Mounts sind sehr kritisch und führen bei unsachgemäßem Einsatz sehr schnell zum Absturz des Betriebssystems. Auch wenn ihre Verwendung auf den ersten Blick bestechend wirkt, ist äußerste Vorsicht angesagt.

3.3.4 Datenträgeraustausch mit dem Macintosh

NetBSD kann auf sehr viele verschiedene Dateisysteme direkt zugreifen. Dazu gehört das (V)FAT-Dateisystem von MS-DOS und Windows genauso wie das von AmigaDOS. Selbst das Lesen von NTFS-Partitionen von Windows 2000 ist möglich.

Generell bei allen Betriebssystemen bereitet es sehr große Schwierigkeiten, wenn Datenträger mit dem Macintosh ausgetauscht werden sollen. Im Normalfall handelt es sich bei solchen Zugriffen um CDs, die aus einem Satzstudio stammen und dort auf einem Macintosh geschrieben wurden. Es reicht aber, wenn man eine Diskette erhält und dringend die Daten benötigt.

Dazu muß man wissen, daß der Macintosh zwar – wenn man es ihm entsprechend sagt – auch FAT-Disketten bearbeiten und ISO9660-CDs schreiben und lesen kann. Gerade letzteres ist aber problematisch. Es gibt einen freien Joliet-Treiber für MacOS 7 und 8 und einen Shareware-Treiber für MacOS 9, die meisten Mac-Anwender wissen das aber nicht und verwenden lieber das native HFS-Format selbst für CDs.

Mit den HFS-Tools aus den Packages (»sysutils/hfsutils«, außerdem wird »devel/libhfs« benötigt) ist nicht nur das Lesen einer Macintosh-CD möglich, auch das Formatieren und Schreiben funktioniert.

Nach der Installation des hfs-Pakets sind im System einige neue Programme vorhanden, die alle mit einem »h« beginnen. Sie heißen beispielsweise *hformat*, *hmount*, *humount*, *hcopy* und *hdir*.

Um eine Macintosh-HFS-Diskette anzulegen, reicht es, *hformat* mit dem Parameter des Diskettenlaufwerks anzugeben (eine Diskette muß natürlich eingelegt sein). Die Diskette muß vorab schon low-level-formatiert sein:

```
#hformat -l "Leerdiskette" /dev/fd0a 0
Volume name is "Leerdiskette"
Volume was created on Wed Dec 11 11:04:21 2002
Volume was last modified on Wed Dec 11 11:04:21 2002
Volume has 1448960 bytes free
```

Die Parameter *-l* für die Angabe des Volume Labels und die abschließende 0 sind optional. Wird *-l* nicht angegeben, erhält die Diskette das Label »Untitled«.

hformat schreibt nur ein HFS-Directory. Die folgenden beiden Zeilen erzeugen eine Macintosh-Diskette von einem unformatierten Datenträger:

```
#dd if=/dev/zero of=/dev/fd0a bs=1k count=1440
1440+0 records in
1440+0 records out
1474560 bytes transferred in 192.033 secs (7678 bytes/sec)
#hformat -l "Test Disk" /dev/fd0a
Volume name is "Test Disk"
Volume was created on Wed Dec 11 11:40:54 2002
Volume was last modified on Wed Dec 11 11:40:54 2002
Volume has 1448960 bytes free
```

Sobald man die Diskette formatiert hat, ist sie auch schon eingebunden. Erhält man eine Fremddiskette, bindet man sie äquivalent ein:

```
#hmount /dev/fd0a
Volume name is "Untitled"
Volume was created on Tue Dec 10 19:17:47 2002
Volume was last modified on Tue Dec 10 19:17:47 2002
Volume has 1448960 bytes free
```

Diese (leere) Diskette wurde vorher auf dem selben Rechner mit *hformat* formatiert. Man kann (muß aber nicht) einen Mountpunkt angeben. Er bringt nicht viel, da die Tools alle von einem internen virtuellen Mountpunkt ausgehen, der durch einen Doppelpunkt gekennzeichnet ist. So kopiert man eine Datei im aktuellen Verzeichnis auf die Mac-Diskette:

```
#hcopy ./Datei :
```

Eine eingebundene HFS-CD ergibt die folgende Ausschrift:

```
#hmount /dev/cd0d
/dev/cd0d: contains 1 HFS partition
Volume name is "OS/2 Warp V4"
Volume was created on Mon Nov  4 17:19:36 1996
Volume was last modified on Mon Nov  4 20:11:03 1996
Volume has 0 bytes free
```

Ein *hdir* ohne weitere Parameter listet den Inhalt des Datenträgers auf. Auf der CD befinden sich zwei Verzeichnisse, wie mit einem Aufruf von *hdir* ermittelt wurde, eines davon hat den folgenden Inhalt (einige Zeilen wurden gekürzt):

```
#hdir PS-Files
f  TEXT/vgrd        0   1070773 Nov  4  1996  Kapitel 1 XP.ps
f  TEXT/vgrd        0   6564348 Nov  4  1996 Kapitel 10 XP.ps
f  TEXT/vgrd        0    803039 Nov  4  1996 Kapitel 12 XP.ps
f  TEXT/vgrd        0   3920575 Nov  4  1996 Kapitel 2 XP.ps
```

```
f  TEXT/vgrd          0   2428642 Nov  4  1996 Kapitel 3 XP.ps
f  TEXT/vgrd          0   3926746 Nov  4  1996 Kapitel 4 XP.ps
f  TEXT/vgrd          0   2835458 Nov  4  1996 Kapitel 5 XP.ps
f  TEXT/vgrd          0  15231317 Nov  4  1996 Kapitel 6 XP.ps
f  TEXT/vgrd          0  11330089 Nov  4  1996 Kapitel 7 XP.ps
```

Eine Datei kopiert man mit *hcopy* vom HFS-Datenträger in das Unix-Dateisystem oder umgekehrt. Das Programm erkennt an der Kennung »:«, ob der HFS-Datenträger Quelle oder Ziel der Aktion ist. Im folgenden Aufruf wird eine Installationsdatei von MacOS 7.5.3 vom lokalen Unix-Dateisystem auf eine HFS-Diskette in deren Hauptverzeichnis kopiert:

```
#hfscopy ./D-System 7.5.3-02of17.part :
```

Wie bei normal gemounteten Datenträgern gilt auch bei den HFS-Tools, daß der Datenträger zuerst gemountet und vor dem Entfernen wieder ausgebunden werden muß. Dazu dient der Befehl *humount*, der immer aufgerufen werden muß, bevor beispielsweise die Diskette entfernt wird.

Der Umgang mit HFS-Datenträgern ist also gewöhnungsbedürftig. Er stellt nicht viel mehr als eine Notlösung für den Umfang mit Macintosh-Datenträgern dar. Im Zweifelsfall ist das besser als nichts und sehr viel mehr als beispielsweise unter Microsoft Windows möglich ist.

Weiterführende Informationen erhalten Sie nach der Installation des Packages aus den Manpages der Programme *hformat*, *hls*, *hcd*, *hattrib*, *hcopy*, *hdel*, *hdir*, *hmkdir*, *hmount*, *hpwd*, *hrename*, *hrmdir*, *humount* und *hvol*. Alle Manpages befinden sich in der Sektion 1.

Wer die HFS-Utilities schon unter FreeBSD verwendet hat, muß sich bei NetBSD etwas umgewöhnen. Das bei FreeBSD vorhandene generische Programm *hfs* wird beim NetBSD-Package nicht installiert. Das ist kein Schaden, da dieses Programm redundant ist (ein Aufruf von *hfs format* entspricht *hformat*).

3.4 Der NetBSD/i386-Bootselektor

von Ignatios Souvatzis

Unaufdringlich und fast unsichtbar verbirgt sich in NetBSD ein Bootselektor für IBM-PC-Kompatible. Im folgenden werden seine Funktionsweise, Installation und Benutzung beschrieben.

Echte Workstations vieler Hersteller, seien es DEC VAX, alte SUNs, Rechner mit Open Firmware, MIPS- und Alpha-Rechner verschiedener Hersteller, aber auch einige Heimcomputer (beispielsweise Amiga ab ROM 36.xxx) bieten schon im Boot-ROM entweder eine tastatur- oder mausgesteuerte Möglichkeit, das Bootgerät (Platte und Partition, teilweise Bandlaufwerk oder Netzwerk) auszuwählen. Es ist zum Beispiel möglich, zwischen verschiedenen Betriebssystemen oder zwischen einer Normal- und einer Testinstallation auszuwählen.

Das PC-BIOS besitzt diese Möglichkeit nur sehr rudimentär; es gibt nur die Möglichkeit, den Codeteil der ersten Festplatte (C:) oder Floppy (A:) oder auch des ersten (teilweise aber nur ATAPI-) CD-ROMs auszuwählen.

Dieser Codeteil auf der Festplatte ist allerdings ziemlich frei in seinen Erweiterungsmöglichkeiten. Alles ist möglich, was das BIOS selber könnte, und das ist genug, um einen echten Bootselektor als primären Bootblock zu schreiben. Zu den Details:

Der Sektor 0 jeder PC-Festplatte enthält eine vierteilige Partitionstabelle und einen Codeteil. Tatsächlich beginnt der Codeteil sogar ganz am Anfang des Sektors, besteht jedoch aus einem Sprungbefehl, der über die Partititionstabelle hinwegreicht. Der Codeteil lädt (normalerweise) den Codeteil des Sektors 0 der als »aktiv« markierten Partition. Ein Betriebssystem hinterläßt in Sektor 0 der eigenen Partition einen Code-Teil, der den Betriebssystemkern, meist auf dem Umweg über einen sekundären Bootlader, lädt.

Ein Bootselektor würde an dieser Stelle auch die Auswahl einer anderen Partition, gegebenfalls sogar einer anderen Platte, erlauben.

Viele Bootselektoren sind relativ aufwendig und laden als erstes ein paar Sektoren des ersten Zylinders, der normalerweise zu keiner Partition gehört; dort befindet sich dann der Rest des Bootselektorcodes. Hierzu gehört beispielsweise OS-BS.

Andere Bootselektoren sind von der Installation her (LILO) oder sogar vom Konzept her (GRUB) kaum vom Bootlader getrennt.

Der NetBSD/i386-Bootselektor dagegen ist wirklich nur ein Bootselektor und überläßt das Laden des Betriebssystems einem Bootlader. Dadurch ist er so kompakt, daß er vollständig in einen Sektor hineinpaßt. Vorteile sind:

- Seine »hinteren« Teile können nicht versehentlich überschrieben werden.
- Er ist immer komplett installiert oder gar nicht vorhanden.
- Weil er nur einen Bootlader (oder einen anderen Bootselektor) startet, kann er auch dazu verwendet werden, ein anderes Betriebssystem als NetBSD zu starten, also beispielsweise Windows oder Linux (letzteres über den Bootlader-Teil von LILO).

Kleine Nachteile sind die etwas spartanische (aber ausreichende) Ausgabe und Steuerung und daß nur Partitionen auf der ersten Festplatte direkt ausgewählt werden können. Zum Auswählen einer Partition auf einer anderen Festplatte muß man zwei Funktionstasten drücken.

Der eingebaute Bootselektor ist seit der Version 1.4 von NetBSD Bestandteil des Betriebssystems; seit NetBSD 1.4.2 ist seine Konfiguration in der jetzigen Form möglich. Vor NetBSD 1.4 gab es als Beigabe zu NetBSD den – ebenfalls frei verfügbaren – OS-BS (zuletzt in der Version 2.0), der jedoch nicht von NetBSD aus, sondern nur von MS-DOS aus konfigurierbar ist.

3.4.1 Anwendung des Bootselektors

Wie schon beschrieben, erlaubt der NetBSD-Bootselektor die Auswahl der Partitionen der Festplatte, auf der er installiert ist. Dies geschieht über die Funktionstasten [F1], [F2], [F3] oder [F4]; der Bootselektor zeigt dazu einen bis zu achtstelligen Namen der bootbaren Partitionen an.

Mit [F5], [F6], [F7], [F8], [F9] oder [F10] ist es möglich, eine andere Festplatte auszuwählen und zwar in der Reihenfolge, wie sie vom BIOS gesehen wird. Wie wählt man dort wiederum eine Partition aus? Nun – dazu muß dort ebenfalls ein Bootselektor installiert sein, dann wird wieder [F1] bis [F4] betätigt.

Außerdem enthält der Bootselektor einen Timeout-Wert und die Taste, die beim Timeout als gedrückt betrachtet werden soll, so daß eine Maschine mit dem Bootselektor trotzdem noch unbeaufsichtigt hochfahren kann.

Installation des Bootselektors

Die Installation oder Konfiguration des Bootselektors geschieht durch *fdisk -B*, gegebenenfalls unter Angabe des (NetBSD-) Namens der Platte.

Als erstes gibt es die Möglichkeit, den Bootselektorcode zu installieren, wenn er noch nicht installiert war. Auf die Meldung »Bootselector not yet installed. Install it now?« wird die entsprechende Auswahl getroffen.

Als zweites ist es möglich, den Bootselektor als aktiv oder inaktiv zu markieren. Dafür zuständig sind die Abfragen »Do you want to deactivate it?« beziehungsweise »Do you want to activate it?«.

Danach hat man die Möglichkeit (falls man den Bootselektor aktiviert hat), die Einstellungen zu ändern. Dafür zuständig ist die Abfrage »Do you want to change its settings?«.

Auf die Antwort [y] (für »yes«) wird nun zunächst die Partitionstabelle ausgegeben, gefolgt von den Bootnamen der Partitionen, beispielsweise:

```
0: NetBSD-1.6
1: Windows
2: <UNUSED>
3: NetBSD-1.4

Change which entry (-1 quits)?
```

Wenn man mit der Tabelle zufrieden ist, gibt man -1 ein, sonst eine Ziffer von 0 bis 3. Jetzt kann der Name eingegeben werden, den die Partition nun besitzen soll. Wenn man direkt die Return-Taste drückt, wird die Partition als unbenutzt (*<UNUSED>*) markiert.

Als nächstes kann man einen Timeoutwert in Sekunden eingeben:

```
        Timeout value:
```

und die Default-Aktion festlegen; im obigen Fall ohne Änderungen ergibt sich:

```
Select the default boot option. Options are:

        0: NetBSD-1.6
        1: Windows
        3: NetBSD-1.4
        4: Harddisk 0
        5: Harddisk 1
        6: Harddisk 2
        7: Harddisk 3
        8: Harddisk 4
        9: Harddisk 5
       10: The first active partition
```

Zum Schluß erfolgt die Sicherheitsabfrage »Update the bootselector?«. Auf die Antwort [y] wird der Bootselektor geschrieben/verändert, sonst bleibt alles beim alten.

Übrigens erlaubt auch das Installationstool *sysinst* die Installation und Konfiguration des Bootselektors schon bei der Installation von NetBSD, die Menüführung entspricht fast genau dem oben beschriebenen.

Ein Trick am Rande: Soll immer von der zweiten Festplatte gebootet werden, muß man einen Bootselektor auf der ersten Platte installieren, der als Defaultaktion die zweite Festplatte enthält. Als Timeoutwert gibt man 1 an.

3.5 Absichern des Systems mit File Flags und Securelevel

von Dr. Lex Wennmacher

Das in Anhang A,1 geschilderte Verfahren, das System mit einem read-only root-Dateisystem abzusichern, erfordert eine Reihe von Klimmzügen, um es ordnungsgemäß ans Laufen zu bekommen. Die Schwierigkeit liegt darin, daß man den Schreibschutz beim Mounten nur für das gesamte Dateisystem einschalten kann. Die von 4.4BSD abgeleiteten Betriebssysteme bieten aber mit den »File Flags« eine interessante Alternative, mit der wesentlich feinkörniger (auf Dateiebene) Schreibschutz einstellen kann.

Die gleich zu erläuternden File Flags funktionieren Hand in Hand mit dem Securelevel. Dieser ist eine vom Kernel verwaltete Sicherheitsstufe, wobei die verschiedenen Stufen folgendes bedeuten:

Securelevel -1: Permanent unsicherer Modus

- Der Kernel läßt keine Erhöhung des Securelevel zu, er läuft ständig im unsicheren Modus.

Securelevel 0: Unsicherer Modus

- Es gibt keine Einschränkungen der Kernelfunktionalität.

Securelevel 1: Sicherer Modus

- System File Flags können gesetzt, aber nicht zurückgesetzt werden.
- Gerätedateien von gemounteten Disks (beispielsweise »/dev/rwd0a«), »/dev/mem« und »/dev/kmem« sind nur noch lesbar, nicht mehr beschreibbar.

Securelevel 2: Hochsicherer Modus. Wie der Sichere Modus, plus:

- Gerätedateien von Disks (ob gemountet oder nicht) sind nur lesbar, aus diesem Grund kann in dieser Sicherheitsstufe *newfs*(8) nicht benutzt werden.
- Gemountete Dateisysteme können nur noch von *read/write* auf *readonly* heruntergefahren werden, die andere Richtung ist nicht möglich.
- Der Systemaufruf *settimeofday(2)* läßt nur das Vorwärtssetzen der Systemzeit zu.
- Die im Kernel gesetzten Filterregeln von *ipf(8)* können nicht mehr verändert werden.
- Benutzer können die für jeden Prozeß einstellbare Kernelvariable, die den Corefile-Namen formatiert (*kern.defcorename*), nicht mehr verändern, nur noch die Änderung des systemweiten Standardwerts durch *root* ist möglich.

Insgesamt bietet die Stufe Securelevel 2 einen sehr sicheren Schutz gegen böswillige Manipulationen des Dateisystems.

Während des Bootvorgangs ist der Securelevel auf 0 (oder -1) eingestellt: alle Manipulationen der Dateisysteme, IP-Filterregeln und so weiter sind uneingeschränkt möglich. Am Ende des Boot-Vorgangs wird normalerweise die Sicherheitsstufe auf 1 erhöht. Auf Wunsch kann man den Securelevel auch auf 2 erhöhen, oder ihn beim Vorliegen besonderer Gründe permanent auf -1 festhalten, was durch die Option

```
#options         INSECURE
```

in der Kernel-Konfigurationsdatei erreicht wird. Dies ist beispielsweise dann erforderlich, wenn man eine Grafikkarte betreiben will, deren Framebuffer vom X-Server via »/dev/mem« beschrieben wird (aus diesem Grund werden die Install-Kernel mit dieser Option kompiliert). Wer solche Grafikkarten betreiben will/muß und dennoch auf die Sicherheit von Securelevel nicht verzichten will, sollte den sogenannten *aperture*-Driver benutzen – ein LKM, der es einem einzigen Prozeß gestattet, den X-Server trotz Securelevel>*0* »/dev/mem« zu beschreiben.

Der Kernel erlaubt es entsprechend privilegierten Prozessen nur, Securelevel zu erhöhen. *root* kann also beispielsweise im laufenden Betrieb Securelevel von 1 auf 2 erhöhen. Das Erniedrigen von Securelevels ist jedoch nur einem einzigen Prozeß erlaubt, nämlich *init*. Dieser Prozeß fährt beim Übergang in den Single-User-Modus den Securelevel auf 0 zurück.

Nun kommen wir zu den bereits erwähnten File Flags. Sie erlauben es unter anderem, ähnlich wie die normalen Unix-Permissions (beispielsweise *-rw-r--r--*) für jede einzelne Datei Zugriffseinschränkungen zu definieren. NetBSD definiert folgende File Flags:

- *schg* (nur root)
- *sappnd* (nur root)
- *arch* (nur root)
- *uchg* (Eigentümer oder root)
- *uappnd* (Eigentümer oder root)
- *opaque* (Eigentümer oder root)
- *nodump* (Eigentümer oder root)

Die File Flags werden mit Hilfe des Kommandos *chflags* gesetzt und wieder gelöscht.

Von besonderem Interesse hier sind die File Flags *schg* und *sappnd*. Eine Datei, auf der das *schg*-Flag gesetzt ist, kann nicht verändert (weder gelöscht, noch umbenannt, noch überschrieben) werden. Ist auf ein Verzeichnis *schg* gesetzt, kann es ebenfalls nicht umbenannt werden, noch können neue Dateien in diesem Directory angelegt oder alte gelöscht oder umbenannt werden. Dateien, auf denen das *sappnd*-Flag gesetzt ist, können zum Lesen geöffnet werden, aber es können an sie nur Daten angehängt werden, das heißt, der bereits bestehende Inhalt ist unveränderbar.

Das Betriebssystem kann nun gegen unberechtigte Veränderungen von Dateien abgesichert werden, indem man beispielsweise auf allen Betriebssystembefehlen (alle Programme in »/bin«, »/sbin«, »/usr/bin«, »/usr/sbin«, »/usr/libexec«), Bibliotheken (in »/usr/lib«) und Konfigurationsdateien (in »/etc«) das *schg*-File-Flag setzt. Auf allen Logdateien (in »/var/log«) wird hingegen das *sappnd*-Flag gesetzt.

In einem typischen Angriffszenario dringt ein Hacker über das Netzwerk in das Opfersystem ein. Normalerweise versucht er als erstes, seine Spuren zu verwischen, danach installiert er Trojaner oder öffnet sich durch Manipulation am System andere Hintertüren. Gehen wir davon aus, daß es dem Hacker gelungen ist, auf der Opfer-Maschine root-Rechte zu erlangen. Das Verwischen seiner Spuren aus den Log-Dateien (beispielsweise aus »/var/log/authlog« oder »/var/log/xferlog«) ist nicht möglich, da diese Dateien mit *sappnd* abgesichert sind. Das Austauschen von »/bin/ps« oder »/bin/ls« gegen eine Version, die seine Identität kaschiert, ist ebenfalls unmöglich, da auf diesen Dateien das *schg*-File-Flag gesetzt ist. Trotz *root*-Rechte können diese Dateien nicht verändert werden. Bliebe dem Hacker also nur die Möglichkeit, die File Flags zurückzusetzen. Das geht aber nur im Single-User-Modus. Dazu müßte der Hacker die Maschine herunterfahren (damit als letzter Schritt *init* den Securelevel wieder auf 0 herunterfährt). Versucht er das allerdings, wird die Netzwerkverbindung unterbrochen (das Netzwerk wird als Teil des Übergangs vom Multi-User-Modus in den Single-User-Modus heruntergefahren). Lediglich von der Konsole aus kommt man dann noch an das System heran. Man sieht zwar, daß die Absicherung eines Systems mit File Flags nicht verhindern kann, daß ein Hacker ins System eindringt, aber seine Schadwirkung wird erheblich eingedämmt.

Man erkennt ebenfalls, daß ein mit File Flags abgesichertes System wesentlich schwieriger zu administrieren ist. Jede kleine Veränderung an einer Konfigurationsdatei erfordert das Herunterfahren des Systems, das Zurücksetzen der File Flags, um die Veränderung vornehmen zu können, gefolgt vom Wiedersetzen des File Flags und einem Übergang in dem Multi-User-Modus. Während dieser Administrationsarbeiten kommt es wegen des notwendigen Weges über den Single-User-Modus zu Service-Unterbrechungen.

Eine weitere wichtige Frage ist, wie man die richtigen File Flags auf einer großen Anzahl von Dateien und Directories setzt und vor allem auch wieder zurücksetzt. Hat man vergessen, auch nur ein File Flag zu löschen, scheitert beispielsweise das nächste Betriebssystem-Upgrade! Die Antwort lautet *mtree*. Dieses Werkzeug wurde dazu entwikkelt, Dateiattribute wie beispielsweise Eigentümer, Gruppe und Protection zu verwalten.

mtree besitzt drei Hauptfunktionen:

- Das Erzeugen einer Definitionsdatei eines bestehenden Verzeichnisbaums (*mtree -c*).
- Vergleichen eines Verzeichnisbaums mit einer vorhandenen Spezifikationsdatei (*mtree*).
- Setzen der Attribute gemäß einer vorhandenen Spezifikationsdatei gemäß einer bestehenden Spezifikationsdatei (*mtree -u*).

Seit NetBSD 1.5 beherrscht *mtree* auch File Flags, und kann zur deren Administration benutzt werden. Die Option *-i* (immutable) von *mtree* bewirkt, daß die *sappnd/schg*-File-Flags von den Dateien gesetzt werden, bei denen dies in der Spezifikationsdatei angegeben ist. Mit *-m* (mutable) wird das genaue Gegenteil erreicht: die File Flags werden wieder zurückgesetzt.

Zur Absicherung des Systems geht man also wie folgt vor:

- Im Normalfall wird das System mit Securelevel 2 betrieben
- root setzt die File Flags *schg* und *sappnd* nach Bedarf. In der Regel werden alle Programme in »/bin«, »/sbin«, »/usr/bin«, »/usr/sbin«, »/usr/libexec«, alle Bibliotheken in »/usr/lib«, alle Konfigurationsdateien und Startscripten in »/etc« und »/etc/rc.d«, »/etc/defaults« und so weiter mit *schg* versehen. Alle Logdateien in »/var/log« dagegen erhalten das *sappnd*-File-Flag. Auch alle Verzeichnisse werden mit *schg* (oder in einigen Fällen mit *sappnd*) abgesichert. Beispiel:

```
# chflags -R schg /bin /sbin /usr/bin /usr/sbin /usr/libexec
# cd /var/log
# chflags sappnd aculog authlog lastlog lpd-errs maillog \
  messages wtmp xdm xferlog
```

- Sind die File Flags alle gesetzt, erzeugt man sich eine Spezifikationsdatei mit *mtree* (es empfiehlt sich, ist aber nicht notwendig, mit Hilfe der *-x* Option *mtree* auf den Verzeichnisbaum des aktuellen Verzeichnisses zu beschränken):

```
# cd /
# mtree -x -c -k flags,type,uname,gname,mode > /etc/mtree/root.secure
# cd /usr
# mtree -x -c -k flags,type,uname,gname,mode > /etc/mtree/usr.secure
etc.
```

- Muß bei der Systemadministration eine mit File Flags abgesicherte Datei verändert werden, fährt man die Maschine in den Single-User-Modus, setzt das File Flag zurück (von der Konsole aus), nimmt die Änderung vor, setzt das File Flag wieder und fährt die Maschine wieder hoch.
- Ist ein größerer Eingriff in das Betriebssystem nötig, der das Zurücksetzen einer größeren Anzahl von File Flags erfordert, fährt man die Maschine in den Single-User-Modus und setzt alle File Flags mit Hilfe von *mtree* zurück:

  ```
  # cd /
  # mtree -u -m /etc/mtree/root.secure
  # cd /usr
  # mtree -u -m /etc/mtree/usr.secure
  ```

  und so weiter .

  Anschließend nimmt man die administrativen Änderungen (an der Konsole) vor und setzt die File Flags wieder alle:

  ```
  # cd /
  # mtree -u -i /etc/mtree/root.secure
  # cd /usr
  # mtree -u -i /etc/mtree/usr.secure
  ```

Zum Abschluß dieses Kapitels noch zwei Bemerkungen:

1. Die Absicherung der Konfigurationsdateien in »/etc« erfordert besonderes Augenmerk. So stellt sich beispielsweise die Frage, ob man »/etc/passwd«, »/etc/pwd.db«, »/etc/spwd.db« und »/etc/master.passwd« mit *schg* oder gar nicht absichert (*sappnd* macht keinen Sinn, da diese Dateien von *pwd_mkdb* immer neu erzeugt werden). Sind sie mit *schg* abgesichert, kann man im laufenden Betrieb keine Benutzer mehr hinzufügen oder Passworte ändern. Sind sie nicht abgesichert, kann ein Hacker natürlich einen Account hinzufügen. Diese Entscheidung muß von Fall zu Fall getroffen werden.
2. Wie man sieht, erschwert diese Absicherung die Systemadministration erheblich, da jede substanzielle Änderung ein Herunterfahren des Systems in den Single-User-Modus erfordert. Diese Methode ist daher nur auf solchen Systemen sinnvoll einzusetzen, auf denen die Bedrohungslage dies rechtfertigt (beispielsweise auf einem Web- oder ftp-Server im Internet).

4 Der Kernel

von Martin Laubach

Der Kernel ist der Dreh- und Angelpunkt eines jeden Betriebssystems. An ihn kommt kein Programm vorbei: er teilt den Prozessen die CPU zu, er verwaltet den (virtuellen) Speicher, er regelt den Zugriff auf die Festplatten, er ermöglicht den Netzzugang und organisiert noch vieles mehr. Durch diese zentrale Rolle trägt das reibungslose Zusammenspiel des Kernels mit der Hardware entscheidend zum Funktionieren des Systems bei.

Warum soll man denn einen eigenen Kernel erzeugen?

NetBSD besitzt einen sogenannten »monolithischen« Kernel, das heißt, die unterstützten Geräte und verfügbaren Optionen sind statisch in die Kernel-Datei eingebunden[1], Änderungen an der Funktionalität sind nur durch ein erneutes Konfigurieren und Neukompilieren des Kernels möglich.

Man kommt also früher oder später um das Erzeugen eines eigenen Kernels nicht herum:

- Manche Hardware wird vom generischen bei der Installation eingespielten Kernel nicht oder nicht optimal unterstützt, weshalb Anpassungen zum einwandfreien Betrieb notwendig sind.
- Durch das genaue Zuschneiden auf die tatsächlich vorhandene Hardware spart ein eigener Kernel viel Speicher, der für andere Dinge als unbenutzte Gerätetreiber gewinnbringender verwendet werden kann.
- Ein schnellerer Bootvorgang ist ein weiterer angenehmer Nebeneffekt: Da nicht nach sowieso nicht vorhandener Hardware gesucht werden muß, wird die Zeit bis zum Hochfahren des Systems drastisch verkürzt.

4.1 Der Kernel-Baukasten

Im Unterschied zu Linux, in dem die Funktionen und Geräte im Kernel durch eine Folge von Frage- und Antwortschritten oder ein baumartiges Menü vom Benutzer erfragt werden, sind für den Bau eines Kernels unter NetBSD nur ein paar Schritte notwendig, die zwar nicht so hübsch präsentiert, dafür aber nicht minder einfach sind. Also keine Angst, es geht keineswegs um Quantenphysik!

[1] Während diese Beschreibung des Kernels durch nachladbare Module in letzter Zeit ein wenig aufgeweicht wurde, ist zum Betrieb der meisten Devices trotzdem eine direkte Einbindung in den Kern notwendig.

4.1.1 Vorbereitungen

Als Voraussetzung für Erzeugung eines neuen Kernels müssen natürlich im Basissystem der C-Compiler und die dazugehörigen Werkzeuge installiert sein, also das Paket »comp.tgz«.

Zur Installation der Quelltexte müssen Sie die Datei »syssrc.tgz« von Ihrem Installationsmedium oder aus dem Internet (zu finden auf *ftp.netbsd.org* oder einem der Mirrors) auf Ihren Rechner kopieren.

Das Archiv kann überall entpackt werden, ohne weiteres auch in Ihrem Homedirectory; root-Berechtigung ist zum Kernelbau (wohl aber dann zur Installation) nicht notwendig. Traditionell werden die Systemquellen jedoch in »/usr/src/« abgelegt oder zumindest ein passender Symlink installiert, da sich manche Werkzeuge darauf verlassen, in »/usr/src/sys« die Kernelquellen zu finden[1].

```
marille [/] % tar -C / -xpzvf /pfad/zu/syssrc.tgz
```

Da diese Datei die Quellen für sämtliche von NetBSD unterstützte Plattformen enthält, könnte dieser Schritt zumindest für eine kleine Kaffeepause gut sein.

Die Kernelquellen bestehen aus plattformunabhängigen Teilen, die für alle Rechnerarchitekturen gleich sind, und spezifischen Teilen, die an die entsprechende Hardware angepaßt sind. Sollten Sie knapp an Festplattenplatz sein, können Sie die Kernelsourcen für Plattformen, die Sie nicht interessieren, einfach löschen: für *i386* beispielsweise können Sie in »/usr/src/sys/arch« alles außer *i386* entfernen[2].

4.1.2 Konfigurationsdateien

Wechseln Sie nun in das Verzeichnis »Plattform«, wobei »Plattform« für den Maschinentyp steht, also etwa *macppc* oder *i386*. Wir werden im folgenden Beispiel von einem PC-System, das ist i386[3], ausgehen. Die Konfiguration für andere Plattformen funktioniert prinzipiell analog, nur einige Konfigurationsdirektiven und natürlich manche plattformspezifische Gerätetreiber unterscheiden sich dabei.

```
marille [/] % cd /usr/src/sys/arch/i386/conf
```

In diesem Verzeichnis befinden sich die Konfigurationdateien für diverse Klassen von Maschinen, zum Beispiel bei i386 *GENERIC*. Das ist der generische, nicht näher auf eine bestimmte Maschine zugeschnittene »Alles-drin«-Kernel, der auch bei der Installa-

---

[1] Es gibt einige wenige Source-Packages die die Kernelsourcen patchen (zum Beispiel »arla« in Kapitel 9.6. Es arbeitet mit dem Link »/sys«, der per Default auf »/usr/src/sys« zeigt.

[2] Manche Plattformen benötigen allerdings auch noch andere Verzeichnisse, beispielsweise verwendet *macppc* auch noch einen Teil aus *powerpc* und *mac68k* aus *m68k*.

[3] Auch moderne Pentium- und Athlon-Prozessoren fallen in die Kategorie i386.

tion von NetBSD von *sysinst* automatisch installiert wird; er ist so konzipiert, daß er ein sehr breites Spektrum möglicher Hardwarekonfigurationen abdeckt. *GENERIC_LAPTOP* ist der generische Kernel mit besonderem Augenmerk auf Laptop-Hardware oder *GENERIC_TINY* für Maschinen mit besonders wenig Hauptspeicher[1].

Diese Konfigurationsdateien beschreiben, was ein Kernel alles enthalten soll – das Format ist dabei recht einfach: Kommentare werden mit »#« eingeleitet, alles danach bis zum Zeilenende wird überlesen. Ansonsten beginnt jede Zeile mit einem Schlüsselwort, gefolgt von dessen Parametern, entweder Gerätedefinitionen von in den Kernel einzubindenden Treibern oder Optionen, die die Funktionalität des Kernels bestimmen.

Um einen eigenen Kernel zu bauen, ist es am sinnvollsten, auf eine bereits vorhandene und funktionierende Kernel-Definition aufzusetzen. Dafür bietet sich natürlich die GENERIC-Definition an. Daher kopieren wir die Datei *GENERIC* auf eine neue Datei um, wir nennen sie hier beispielhaft *MARILLE* – als Namen sollte man etwas wählen, das den Kern beschreibt, also beispielsweise den Hostnamen (»MYFILESERVER«), oder die allgemeine Klasse von Maschinen (»ALDI_PC«). Traditionsgemäß steht ein Kernelname immer in Großbuchstaben.

```
marille [...sys/arch/i386/conf] % cp GENERIC MARILLE
```

Nun können wir gleich in medias res gehen und die Datei editieren:

```
marille [...sys/arch/i386/conf] % vi MARILLE
```

Sie können nun gleich den ersten Schritt tun und den Kern auf den tatsächlichen Prozessor im Rechner optimieren. Aus den Zeilen

```
# CPU support.  At least one is REQUIRED.
options         I386_CPU
options         I486_CPU
options         I586_CPU
options         I686_CPU

# CPU-related options
options         MATH_EMULATE    # floating point emulation

# eliminate delay no-ops in I/O; recommended on all but very old machines
#options        DUMMY_NOPS
```

[1] Der Versuch, einen Kernel auf solch einer minimalen Maschine zu bauen, kann allerdings durchaus mehrere Stunden in Anspruch nehmen. Besser ist es allemal, eine schnelle Maschine zum Kernelbau zu verwenden und die Datei dann auf die Zielmaschine zu kopieren.

kommentieren Sie die nicht benötigten CPU-Optionen aus oder löschen Sie sie. Auskommentieren ist allerdings besser, denn das kann man bei Bedarf leichter rückgängig machen :-). Außerdem können Sie bei einer halbwegs »modernen« Maschine (alles ab 486DX oder mit vorhandenem x87-Coprozessor) *MATH_EMULATE* ebenfalls auskommentieren, da das dann der Floating-Point-Teil der CPU übernimmt. Und, wie es der Kommentar empfiehlt, *DUMMY_NOPS* einschalten (das heißt, das Kommentarzeichen entfernen).

4.1.3 Kompilation

Speichern Sie die Datei nun ab und beenden Sie den Editor. Zuerst muß aus der Konfigurationsdatei ein Verzeichnis generiert werden, das die notwendige Informationen zum Kernelbau enthält. Dazu rufen Sie *config(8)* mit dem Namen Ihrer Datei auf:

```
marille [...sys/arch/i386/conf] % config MARILLE
Don't forget to run "make depend"
```

Hiermit wurde in ein neues Verzeichnis mit dem Namen Ihres Kernels angelegt – in unserem Fall also »MARILLE« – das die zum Bau eines neuen Kernels notwendigen Dateien enthält. Das sind einerseits generierte *Headerdateien* für den C-Compiler, die die Optionen im Configfile widerspiegeln, andererseits auch ein *Makefile*, das den Kernelbauprozeß fast vollständig automatisiert.

Wechseln Sie in das Verzeichnis und folgen Sie dem Ratschlag von *config*:

```
marille [...sys/arch/i386/conf] % cd ../../compile/MARILLE
marille [...arch/i386/compile/MARILLE] % make depend
```

Dieser Schritt analysierte, welche Dateien neu kompiliert werden müssen.[1]

Jetzt ist alles bereit, den Kern zu bauen:

```
marille [...arch/i386/compile/MARILLE] % make
```

Das kann nun abhängig von der Hardware zwischen Stunden (486/33, Amiga oder VAX) und ein paar Minuten (P-III 800 oder neuere Sun) dauern. Am Ende des Compilerlaufs sollte etwas in der Art

[1] Bei einem erstmaligen Bau des Kernels (das heißt, das Verzeichnis »../../compile/MACHINE« wurde erzeugt) ist der Schritt *make depend* eigentlich nicht notwendig, aber konsequentes *make depend* nach jedem Konfigurationslauf verhindert, daß geänderte Konfigurationen nicht in den erzeugten Kernel durchschlagen.

```
ld -Ttext c0100000 -e start -X -o netbsd ${SYSTEM_OBJ} vers.o
text    data    bss     dec     hex     filename
1082487 63992   168160  1314639 140f4f  netbsd
```

auf Ihrem Terminal erscheinen.

Damit ist der neue Kernel fertig kompiliert. Sie könnten ihn nun ins Root-Verzeichnis kopieren und gleich mal ausprobieren:

```
marille [...arch/i386/compile/MARILLE] % mv /netbsd /onetbsd
marille [...arch/i386/compile/MARILLE] % cp netbsd /
marille [...arch/i386/compile/MARILLE] % shutdown -r now
```

Im Unterschied zu Linux ist hier kein Aufruf von Lilo oder sonstigem notwendig – der Bootblock von NetBSD kann den Kernel ohne weiteres Zutun direkt von der NetBSD-Partition lesen.

Es hat sich übrigens als sinnvoll herausgestellt, den derzeit aktiven Kern nicht einfach zu überschreiben, sondern wie oben gezeigt auf einen neuen Namen umzubenennen und erst dann den neuen Kernel zu installieren. Damit kann man im Notfall, sollte der neue Kernel aus irgendwelchen Gründen nicht ordnungsgemäß funktionieren, jederzeit über den Bootloader wieder den alten Kernel laden – das wäre ungleich schwieriger, wenn man ihn gelöscht oder überschrieben hätte.

Weiter ist es eine gute Idee, den GENERIC-Kernel immer als */netbsd.generic* aufzuheben, um selbst bei einem Motherboard- oder Prozessorwechsel einen Kernel zur Verfügung zu haben, der garantiert funktioniert.

Am Beginn des Boot-Prozesses sollte nun eine Zeile der Form

```
NetBSD 1.6 (MARILLE) #0: Thu May 30 23:57:13 CEST 2002
    mjl@marille.example:/usr/src/sys/arch/i386/compile/MARILLE
```

auftauchen und uns damit bestätigen, daß tatsächlich unser gerade neu erzeugter Kernel verwendet wird. Diese Zeile beweist, daß die Konfigurationsdatei *MARILLE* verwendet wurde, das Datum ist die Uhrzeit des Kernelbaus, und »#0« zeigt, daß es der erste Kernel ist, der aus dieser Konfigurationsdatei erzeugt wurde.

Nach dem erfolgreichen Erzeugen (und Installieren) des neuen Kernels wird das Verzeichnis »/usr/src/sys/arch/i386/compile/MARILLE« nicht mehr benötigt und kann gelöscht werden. Wenn Sie allerdings planen, des öfteren neue Kernel für Ihre Maschine zu bauen, etwa um den Effekt der verschiedenen Einstellungen auszuprobieren, so sollten Sie das Verzeichnis aufbewahren – weitere Übersetzungsläufe sind dann wesentlich schneller, da nur die veränderten Komponenten neu kompiliert werden müssen.

4.1.4 Im Notfall

Sollte sich der neue Kernel wirklich als nicht verwendbar herausstellen (zum Beispiel weil ein wichtiger Gerätetreiber, etwa die Tastatur, vergessen wurde), können Sie – vorausgesetzt, Sie haben den alten Kernel aufgehoben – vom Boot-Prompt diesen Kernel starten. Dazu unterbrechen Sie den Countdown beim Starten durch einen Druck auf die Leertaste.

```
>> NetBSD BIOS Boot, revision 2.2
>> (user@buildhost, builddate)
>> Memory: 637/15360 k
Use hd1a:netbsd to boot sd0 when wd0 is also installed
Press return to boot now, any other key for boot menu
booting wd0a:netbsd - starting in 5
```

Nun können Sie den alten Kernel durch

```
boot onetbsd
```

laden.

4.1.5 Struktur der Hardware

Wie bewiesen, müssen Sie, um einen Kernel korrekt zu konfigurieren, einiges über den Aufbau Ihres Computers wissen – vom Prozessor über eingebaute Bussysteme wie ISA oder PCI bis zu den diversen Steckkarten für Netzwerk oder Video.

Woher aber nun die genaue Topologie des Rechners erraten? Netterweise nimmt uns der Kernel einen Teil dieser Arbeit ab – der GENERIC-Kernel enthält ja einen Großteil aller möglichen Gerätetreiber und wird daher die meisten Devices selbständig erkennen. Während des Bootens wird jedes gefundene Gerät auf der Konsole mitgeloggt. Mit dem Befehl *dmesg* (oder bei gemountetem *kernfs* in der Datei »/kern/mgsbuf«) können Sie sich jederzeit die Bootmeldungen ansehen, die auszugsweise auf einem Intel-PC so aussehen könnten:

```
NetBSD 1.6 (MARILLE) #0: Thu May 30 23:57:13 CEST 2002
    mjl@marille.example:/usr/src/sys/arch/i386/compile/MARILLE
cpu0: Intel Pentium (P54C) (586-class), 133.66 MHz
total memory = 65152 KB
avail memory = 58304 KB
using 840 buffers containing 3360 KB of memory
BIOS32 rev. 0 found at 0xfb860
mainbus0 (root)
pci0 at mainbus0 bus 0: configuration mode 1
pci0: i/o space, memory space enabled
pchb0 at pci0 dev 0 function 0
```

```
pchb0: Silicon Integrated System product 0x5511 (rev. 0x00)
pcib0 at pci0 dev 1 function 0
pcib0: Silicon Integrated System 85C503 or 5597/5598 ISA bridge (rev. 0x01)
Silicon Integrated System 5597/5598 IDE controller (IDE mass storage, interface
0x8a, revision 0x05) at pci0 dev 1 function 1 not configured
de0 at pci0 dev 15 function 0
de0: interrupting at irq 9
de0: 21140A [10-100Mb/s] pass 2.2
de0: address 00:80:c8:94:38:c8
ahc0 at pci0 dev 17 function 0
ahc0: interrupting at irq 11
ahc0: aic7880 Wide Channel A, SCSI Id=7, 16/255 SCBs
scsibus0 at ahc0 channel 0: 16 targets, 8 luns per target
isa0 at pcib0
ne1 at isa0 port 0x300-0x31f irq 10
ne1: NE2000 Ethernet
ne1: Ethernet address 08:00:21:97:00:00
scsibus0: waiting 2 seconds for devices to settle...
ahc0: target 2 using 16bit transfers
ahc0: target 2 synchronous at 20.0MHz, offset = 0x8
ahc0: target 2 using tagged queuing
sd2 at scsibus0 target 2 lun 0: <IBM, DDYS-T09170N, S96H> SCSI3 0/direct fixed
sd2: 8748 MB, 15110 cyl, 3 head, 395 sec, 512 bytes/sect x 17916240 sectors
...
```

Diese Bootmeldungen entsprechen der folgenden Grafik. Wie zu sehen ist, wird jedes Gerät im Format *device at bus* erwähnt.

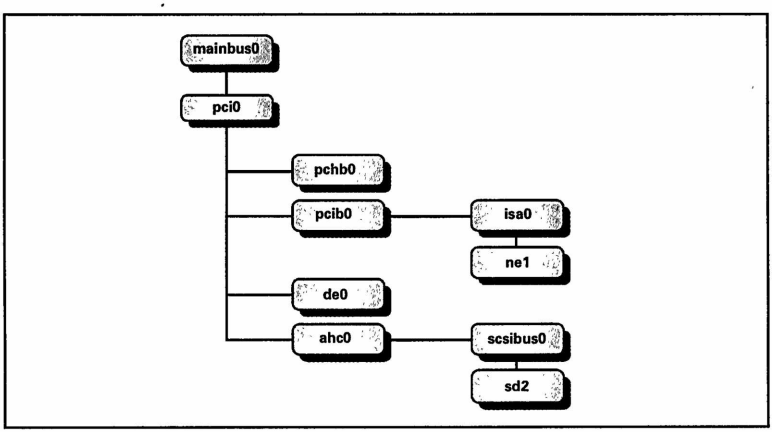


Abb. 4.1: Die Verknüpfung der Geräte mit dem Bus

Auf einem Macintosh mit 680x0-Hardware sehen die Bootmeldungen sehr ähnlich aus, auch wenn sich naturgemäß die Namen der Geräte von der Intel-Plattform unterscheiden:

```
NetBSD 1.6 (GENERIC) #0: Fri Nov  1 05:23:18 CET 2002
    root@desktop:/usr/src/sys/arch/mac68k/compile/GENERIC
Apple Macintosh Quadra 700  (68040)
cpu: delay factor 800
total memory = 20480 KB
avail memory = 15076 KB
using 281 buffers containing 1124 KB of memory
mrg: 'Mac IIci/Q700 ROMs' ROM glue, tracing off, debug off, silent traps
mainbus0 (root)
obio0 at mainbus0
esp0 at obio0 (quick): address 0x88f000: NCR53C96, 25MHz, SCSI ID 7
scsibus0 at esp0: 8 targets, 8 luns per target
adb0 at obio0
asc0 at obio0: Apple Sound Chip
intvid0 at obio0 @ f9000e00: DAFB video subsystem, monitor sense 1
intvid0: 832 x 624, 256 color
macfb0 at intvid0
wsdisplay0 at macfb0 (kbdmux ignored): console (std, vt100 emulation)
sn0 at obio0: integrated Ethernet adapter
sn0: Ethernet address 08:00:07:9c:a4:16
iwm0 at obio0: Apple GCR floppy disk controller
fd0 at iwm0 drive 0: (drive empty)
zsc0 at obio0 chip type 0
zsc0 channel 0: d_speed   9600 DCD clk 0 CTS clk 0
zstty0 at zsc0 channel 0
zsc0 channel 1: d_speed   9600 DCD clk 0 CTS clk 0
zstty1 at zsc0 channel 1
nubus0 at mainbus0
ae0 at nubus0 slot d: MacCon Ethernet, 64KB memory
ae0: Ethernet address 00:00:94:07:c4:c4
fpu0 at mainbus0 (mc68040)
scsibus0: waiting 2 seconds for devices to settle...
sd0 at scsibus0 target 0 lun 0: <IBM, DCAS-34330, S65A> SCSI2 0/direct fixed
sd0: 4134 MB, 8205 cyl, 6 head, 171 sec, 512 bytes/sect x 8467200 sectors
sd0: sync (160.0ns offset 15), 8-bit (6.250MB/s) transfers, tagged queueing
cd0 at scsibus0 target 6 lun 0: <PIONEER, DVD-ROM DVD-303, 1.06> SCSI2 5/cdrom
...
```

Daraus läßt sich recht leicht die tatsächlich vorhandene Hardware rekonstruieren. Diese Arbeit läßt sich allerdings auch recht gut automatisieren – dafür sind Computer ja da. Im NetBSD-Package-System befindet sich unter der Kategorie *sysutils* das Paket *adjustkernel*, das genau das tut und aus den Bootmeldungen und der GENERIC-Datei eine

Konfigurationsdatei generiert, die nur noch die auch tatsächlich vorkommenden Geräte enthält.

Es kann auch vorkommen, daß der Kernel ein angeschlossenes Gerät zwar erkennt, aber keinen passenden Treiber dafür findet. In den Bootmeldungen sähe dieser Fall dann etwa so aus:

```
Brooktree Bt848 Video Capture (video multimedia, revision 0x12) at pci2
dev 4 function 0 not configured
```

In diesem Fall muß man den entsprechenden Treiber in der Kernel-Konfigurationsdatei einschalten – oder man hat Pech und es gibt tatsächlich keine Unterstützung für die Karte.

Zusammenfassend lassen sich die Konfigurationsarbeiten an einem eigenen Kernel in drei prinzipielle Kategorien einordnen:

- Gerätetreiber für Hardware werden ein- oder ausgeschaltet (zum Beispiel kann das Audio-Subsystem auf Maschinen ohne Soundkarte entfernt werden).
- Optionen im Kernel werden ein- oder ausgeschaltet (zum Beispiel kann die Funktionalität *NFS-Client* abgeschaltet werden).
- Diverse Tuningarbeiten an Kernelparametern (wie etwa die Festlegung der Größe interner Tabellen).

4.2 Die Konfigurationsdatei

Eine Kernel-Konfigurationsdatei besteht also, wie Sie ja schon gesehen haben, im wesentlichen aus drei unterschiedlichen Arten von Anweisungen:

- (Hoffentlich hilfreichen) Kommentaren, beginnend mit »#« bis zum Ende der Zeile. Kommentare dienen nur der Orientierungshilfe und werden bei der Generierung des Kernels überlesen. Die Kommentare im *GENERIC* geben meist kurze Hinweise, welcher Treiber für welche Geräteklasse paßt und was die diversen Optionen bewirken. Kommentare werden auch gerne dafür benutzt, nicht benötigte Optionen oder Treiber abzuschalten, ohne die Einträge tatsächlich zu löschen.

- Optionen, die die Funktionen des Kernels verändern oder beeinflussen. Sie beginnen (meistens) mit dem Schlüsselwort *options*, gefolgt von Parametern zu der Option. Diese Parameter sind entweder ein Wort, wie in *options INET*, die eine Funktion ein- (bei vorhandener Option) oder ausschalten (bei entfernter oder auskommentierter Option), oder aber auch eine Wertzuweisung wie in *options RTC_OFFSET=0*, die einen vorhandenen, defaultmäßig angenommenen Wert verändert.

- Devices, die vorhandene Hardware beschreiben. Sie haben immer die Form *device at bus*, möglicherweise gefolgt von weiteren benötigten Parametern, um die Hardware korrekt einbinden zu können.

Sollten beim Editieren der Konfigurationsdatei gravierende Fehler passiert sein, bemängelt *config* das gleich. Eines der am häufigsten vorkommenden Probleme ist das Abschalten eines Geräts, das für andere Devices benötigt wird. Wird zum Beispiel der ISA-Bus

entfernt (es wurden alle Zeilen der Form *isa0 at foo* entfernt), wird das durch die Meldung

```
MARILLE:331: npx0 at isa? is orphaned
(nothing matching isa? declared)
```

angemerkt. So ein Fehler muß korrigiert werden, bevor ein Kernel gebaut werden kann.

Verdrahtung und Wildcards

Bei der Definition der Gerätestruktur werden oft, wie Ihnen beim Durchblättern des GENERIC-Configfiles sicher schon aufgefallen ist, die Wildcards »*« und »?« verwendet. Ähnlich wie bei Operationen in der Shell kann damit eine Menge Arbeit gespart werden.

Ein Gerät kann grundsätzlich in der Form

```
foo0 at bus0
```

definiert werden. Wird dieses Statement so in der Konfigurationsdatei erwähnt, wird zur Laufzeit exakt nur dieses eine Gerät an diesem einen Bus entdeckt. Oft ist aber der genaue Attachment-Punkt nicht weiter wichtig, weshalb obige Form abgewandelt werden kann zu

```
foo0 at bus?
```

Damit wird nun ein foo-Gerät gefunden, egal, an welcher Instanz des Bustyps es nun angeschlossen ist. Allerdings wird damit natürlich nur ein foo-Gerät eingebunden (man kann ja schlecht mehreren Geräten den gleichen Namen geben, nämlich *foo0*). Daher gibt es auch noch die auf der linken Seite erweiterte Form

```
foo* at bus?
```

die nun auch noch beliebig viele Instanzen eines foo-Geräts einbindet. Die Gerätenummern werden dabei aufsteigend, beginnend mit der ersten nicht fixen Nummer vergeben. So führt zum Beispiel der folgende Ausschnitt aus dem GENERIC-File zusammen mit nur einer PCI-NE2000-kompatiblen Karte zu dem etwas überraschenden Ergebnis, daß das erste (und einzige) Gerät als *ne2* eingebunden wird.

```
ne0   at isa? port 0x280 irq 9              # NE[12]000 ethernet cards
ne1   at isa? port 0x300 irq 10
ne*   at pci? dev ? function ?              # NE2000-compatible Ethernet
```

Wenn Sie nun die Einträge für *ne0* und *ne1* entfernen, da ja sowieso keines dieser Geräte vorhanden ist, wird bei Ihrem selbstkompilierten Kernel dieses Gerät, das unter GENERIC als *ne2* bekannt war, plötzlich als *ne0* erscheinen.

Es kann auch sinnvoll sein, statt Wildcards explizite Gerätenummern zu vergeben, zum Beispiel bei Harddisks, denn hier ist es eher unerwünscht, daß durch das Hinzufügen oder Entfernen von Platten die bereits vorhandenen Disks auf einmal die Gerätenummer wechseln.

Nehmen wir beispielsweise ein System mit zwei SCSI-Platten. Im GENERIC-Kernel wird dies durch das Statement

```
sd*   at scsibus? target ? lun ?        # SCSI disk drives
```

abgedeckt und die Platten werden als *sd0* und *sd1* (in der Reihenfolge ihrer SCSI-IDs) erscheinen. Wird nun auf *sd1* ein Filesystem eingerichtet und mittels Eintrag in »/etc/fstab« automatisch gemountet, funktioniert alles prächtig, so lange sich an der Hardwaretopologie nichts ändert. Wird *sd0* entfernt oder vor *sd1* eine neue Platte angeschlossen, verschiebt sich *sd1* plötzlich auf *sd0* beziehungsweise *sd2*. Das läßt sich durch explizite Verdrahtung der Hardwaretopologie erreichen, die Konfiguration wird dann als »wired down« bezeichnet, die den Platten explizite Devicenummern zuweist. Siehe auch Seite 195.

Plattformunabhängige Optionen

Im folgenden beschäftigen wir uns näher mit den möglichen Optionen und Gerätetreibern. Durch die schiere Menge der verfügbaren Treiber ist allerdings eine erschöpfende Abhandlung so gut wie unmöglich – es sei hier auf die mitgelieferte Dokumentation verwiesen (*man 4 options*, sowie für so gut wie jeden Gerätetreiber *man 4 device*, also zum Beispiel *man 4 ahc*).

Standardoptionen

```
maxusers    32                  # estimated number of users
```

Anders als der Name vermuten läßt, beschränkt oder limitiert diese Einstellung nicht die Anzahl der zugelassenen Benutzer, sondern es handelt sich vielmehr um eine Schätzung der wahrscheinlich gleichzeitig aktiven Anwender, die gebraucht wird, um diverse interne Tabellen optimal anzulegen.

Unter anderen wird zum Beispiel die Prozeßtabelle, die ja die Maximalanzahl der gleichzeitig laufenden Prozesse beschränkt, direkt von *maxusers* abgeleitet. Auch die Anzahl der allokierten Dateideskriptoren wird aus *maxusers* berechnet. Ein zu hoch gewählter Wert wird daher unnötigerweise zuviel Speicher reservieren, ein zu niedriger Wert dagegen wird die Benutzbarkeit des Systems stark einschränken, neu gestartete Prozesse werden gleich mit »Proc Table full« oder »Cannot fork« terminiert. Der Wert 32 ist für die meisten Anwendungen ein guter Durchschnitt.

```
options   INCLUDE_CONFIG_FILE          # embed config file in kernel binary
```

Ist diese Option eingeschaltet, wird die Kernel-Konfigurationsdatei in das entstehende Kernel-Image eingebunden. Dies geschieht mit dem Hintergedanken, die genaue Kernel-Konfiguration auch bei Verlust der Konfigurationsdatei wiederherstellen zu können. Man kann die Daten dann mit

```
strings netbsd | sed -n 's/^_CFG_//p' | unvis
```

rekonstruieren.

```
options   UCONSOLE             # users can use TIOCCONS (for xconsole)
```

Um als normaler User Zugriff auf die Konsole zu erlangen, muß diese Option gesetzt sein. Das ist zum Beispiel bei einer Workstation, die X11 verwendet, notwendig. Wenn bei Ihnen X11 nicht im Einsatz ist, können Sie ruhigen Gewissens auch *UCONSOLE* auskommentieren.

```
options     INSECURE           # disable kernel security levels
```

Normalerweise unterscheidet der Kernel mehrere Sicherheitsebenen (siehe *init(8)*). Dabei gibt es Ebene 0, den »unsicheren Modus«, in dem Zugriff auch auf unveränderliche Dateien (siehe *chflags(1)*) erlaubt ist und alle Geräte zugänglich sind. Das System läuft früh im Bootprozeß und im Single-User-Modus auf Ebene 0. Des weiteren gibt es Ebene 1, den »sicheren Modus«. Das ist der normale Sicherheitslevel im Multi-User-Modus. Hier können als unveränderlich markierte Dateien nicht mehr modifiziert werden und bestimmte Geräte wie etwa */dev/kmem* und Disks, die gerade gemountet sind, sind nicht mehr frei zugänglich. Ebene 2, der »hochsichere Modus«, wird nur in bestimmten Umgebungen eingesetzt, da er noch weitere Beschränkungen auferlegt, die im normalen Betrieb oft hinderlich sind.

Und schließlich gibt es auch noch Ebene –1, den »permanent unsicheren Modus« – das System läuft immer mit der Sicherheit der Ebene 0. Genau das wird durch den Schalter *INSECURE* bewirkt. Sie sollten *INSECURE* nur verwenden, wenn Sie planen, X11 einzusetzen. Sonst ist es üblicherweise nicht notwendig, ein System unter *INSECURE* laufen zu lassen. Auf Maschinen wie Firewalls oder Routern sollten Sie *INSECURE* keinesfalls setzen!

```
options   RTC_OFFSET=0      # hardware clock is this many mins. west of GMT
```

Bestimmt, in welcher Zeitzone die Hardwareuhr der Maschine läuft. Normalerweise hält NetBSD die Uhr des Rechners in UTC (aka GMT), aber wenn Sie ein anderes Betriebssystem parallel verwenden, das die Hardwareuhr in lokaler Zeit erwartet (Windows), können Sie die Vorstellung NetBSDs vom aktuellen Datum damit entsprechend korrigieren.

```
options     NTP             # NTP phase/frequency locked loop
```

Diese Option erlaubt *ntpd(8)* eine noch größere Genauigkeit in der Uhrensynchronisation mit einem entfernten System. *ntpd* wird auch auf Systemen ohne diese Option laufen, kann dann allerdings nicht so genau arbeiten und wird einige Millisekunden abweichen.

```
options     KTRACE          # system call tracing via ktrace(1)
```

Erlaubt das Verfolgen von Prozessen durch *ktrace(1)*. Siehe auch die dazugehörige Manpage.

```
options     LKM             # loadable kernel modules
```

Erlaubt das Nachladen von LKMs (Loadable Kernel Modules). Da es aber bei NetBSD kaum nennenswerte LKMs gibt, ist diese Option meist überflüssig. Auf sicherheitsrelevanten Maschinen (Firewall oder Router) sollten Sie LKMs auf jeden Fall abschalten.

System V Interprozeßkommunikation

Die folgenden Schalter sind für die Konfiguration von System V IPC (Interprocess Communication). Sie sollten Sie nicht auskommentieren, denn einige Programme wie zum Beispiel PostgreSQL verwenden Shared Memory oder Semaphore für die interne Kommunikation. Die auskommentierten Zeilen zeigen die Standardeinstellungen der Sys-V-IPC-Mechanismen, sie sollten bei Bedarf nach oben angepaßt werden.

```
options         SYSVMSG         # System V-like message queues
options         SYSVSEM         # System V-like semaphores
#options        SEMMNI=10       # number of semaphore identifiers
#options        SEMMNS=60       # number of semaphores in system
#options        SEMUME=10       # max number of undo entries per process
#options        SEMMNU=30       # number of undo structures in system
options         SYSVSHM         # System V-like memory sharing
#options        SHMMAXPGS=1024  # 1024 pages is the default
```

Debugging

Auch die Programmierer der Kernel-Komponenten sind vor Fehlern nicht gefeit – darum gibt es eine Handvoll Optionen, die zusätzliche Überprüfungen des Systemzustands zur Laufzeit aktivieren.

```
# Diagnostic/debugging support options
#options        DIAGNOSTIC      # expensive kernel consistency checks
```

DIAGNOSTIC schaltet interne Konsistenzprüfungen zu, die bei einem entdeckten Fehler zu einer Panic führen. Sie sollten nur aktiviert werden, wenn sie tatsächlich benötigt werden, da dadurch die Performance des Systems um bis zu fünfzehn Prozent reduziert wird.

```
#options    DEBUG           # expensive debugging checks/support
```

Durch diesen Schalter wird generell der Debugging-Modus aktiviert und viele Teile des Kernels können nun zusätzliche Informationen auswerfen. Auch hier gilt: Nicht ohne Anlaß verwenden, da auch *DEBUG* die Performance drückt!

```
options     DDB                 # in-kernel debugger
options     DDB_HISTORY_SIZE=512    # enable history editing in DDB
```

Um ein System bei einer Panic zu diagnostizieren zu können, wird der Kernel-Debugger DDB eingebunden, mit dem man die internen Datenstrukturen des laufenden Kernels anzeigen lassen und manipulieren kann. Sie sollten diesen Schalter bei einem Produktionssystem nicht verwenden, da bei einem Absturz die Maschine automatisch im Debugger landet und auf interaktive Eingaben wartet, anstatt einfach neu zu booten[1].

```
#options DDB_ONPANIC=0
```

Ist der Wert auf 1 gesetzt, fällt die Maschine bei einer Panic in den Debugger (siehe oben); ist die Einstellung dagegen 0, wird automatisch rebootet. Diese Option kann auch zur Laufzeit mit *sysctl(8)* durch die Variable *ddb.onpanic* verändert werden.

```
#makeoptions    DEBUG="-g"          # compile full symbol table
```

Bestimmt, ob ein Kernel mit Symbolinformation gebaut wird. Ist dieser Schalter gesetzt, wird zusätzlich zum normalen Kernel *netbsd* ein Kernel mit Symbolen *netbsd.gdb*, der zur Analyse von Crash-Dumps benötigt wird, erzeugt. Im Normalfall braucht man diese Option nicht.

```
#options MSGBUFSIZE=8192
```

Damit wird die Größe des Ausgabepuffers des Kernels bestimmt. Das ist der Bereich, der vom Befehl *dmesg* ausgegeben wird. Bei Systemen mit vielen Steckkarten und internen Geräten kann es vorkommen, daß die Standardgröße nicht ausreicht und der Anfang der Bootmeldungen verloren geht. Sollte das passieren, setzt man *MSGBUFSIZE* auf einen angemessenen Wert.

---

[1] Ein nicht mehr reagierendes System am DDB-Prompt am anderen Ende der Stadt kann einem den ganzen Tag verderben.

```
options         EISAVERBOSE       # verbose EISA device autoconfig messages
options         MIIVERBOSE        # verbose PHY autoconfig messages
options         PCIVERBOSE        # verbose PCI device autoconfig messages
#options        PCI_CONFIG_DUMP   # verbosely dump PCI config space
#options        PCMCIAVERBOSE     # verbose PCMCIA configuration messages
options         SCSIVERBOSE       # human readable SCSI error messages
options         USBVERBOSE        # verbose USB device autoconfig messages
#options        PNPBIOSVERBOSE    # verbose PnP BIOS messages
#options        PNPBIOSDEBUG      # more fulsome PnP BIOS debugging messages
options         I2OVERBOSE        # verbose I2O driver messages
options         MCAVERBOSE        # verbose MCA device autoconfig messages
```

Diese Optionen übersetzen kryptische, kurze Meldungen der erwähnten Subsysteme in lesbare Form. Allerdings werden dadurch recht große Übersetzungstabellen in den Kernel eingebunden, so daß man diese Schalter nur setzen sollte, wenn ein paar weitere Kilobyte nicht ins Gewicht fallen.

```
#options MAXDSIZ =(1*1024*1024*1024)
#options DFLSSIZ=(2*1024*1024)
#options MAXSSIZ=(32*1024*1024)
```

Diese Variablen setzen die Maximalgrößen des Datensegments eines Prozesses (*Max Data Segment Size*), die Standardgröße des Stacks (*Default Stack Size*) sowie die Maximalgröße des Stacks (*Max Stack Size*).

Plattformspezifische Optionen (i386)

Die folgenden Optionen beschreiben die Hardware, auf der der Kernel laufen soll. Diese Schalter wurden ja bereits im ersten Beispiel vorgestellt. Man optimiert mit ihnen den Kernel-Code auf einen spezifischen Prozessor und blendet nicht verwendete Subsysteme (die Floatingpoint-Emulation) aus.

Prozessorlastiges

```
# CPU support.  At least one is REQUIRED.
options         I386_CPU
options         I486_CPU
options         I586_CPU
options         I686_CPU
# CPU-related options.
options         MATH_EMULATE      # floating point emulation
# eliminate delay no-ops in I/O; recommended on all but very old machines
options         MTRR              # memory-type range register syscall support
options         PERFCTRS          # performance-monitoring counters support
```

```
# Math Coprocessor support
npx0        at isa? port 0xf0 irq 13     # x86 math coprocessor
```

Der Treiber für den mathematischen Coprozessor wird immer benötigt, egal, ob nun tatsächlich eine physische Floating-Point-Unit vorhanden ist oder die Emulation *MATH_EMULATE* die Arbeit erledigt. Ein Kernel ohne *npx* ist nicht lauffähig.

```
#options     DUMMY_NOPS
```

Entfernt NOPs (No-Operation-) Instruktionen, die bei älteren Maschinen zu Timingzwecken verwendet werden. Auf halbwegs moderner Hardware sollten diese NOPs nicht mehr nötig sein, ihre Entfernung beschleunigt kritische Routinen im Kernel.

```
#options CYRIX_CACHE_WORKS
```

Betrifft nur die Cyrix 486DLC-CPU und schaltet den Hold-Flush-Modus des Prozessors ein. Defaultmäßig ist diese Option abgeschaltet, da manche Motherboards mit diesem Modus nicht zurechtkommen.

```
#options CYRIX_CACHE_REALLY_WORKS
```

Betrifft nur die Cyrix 486DLC-CPU und schaltet den Write-back-Cache des Prozessors zu. Wie oben ist auch dieser Schalter normalerweise deaktiviert, da es mit manchen Motherboards zu Problemen kommt.

Speicherndes

Die folgenden Optionen, mit denen die Speichergröße explizit gesetzt wird, sind eigentlich nie notwendig. Sie dienen nur als Workaround für sehr alte, fehlerhafte BIOS-Routinen, die die Menge vorhandenen Speichers falsch erkennen oder weitergeben. Sie können diese Schalter natürlich auch verwenden, um eine Maschine mit wenig RAM zu simulieren.

```
#options     REALBASEMEM=639      # size of base memory (in KB)
#options     REALEXTMEM=15360     # size of extended memory (in KB)
```

Sparsames

APM (Advanced Power Management) versetzt die Maschine in Schlaf- oder Suspendzustand. Dafür muß das Device *apm* eingebunden werden.

```
#apm0        at mainbus0              # Advanced power management
```

Da sich die BIOS-Hersteller nicht ganz einig waren, welche Features wie – wenn überhaupt – implementiert werden sollen, gibt es noch die folgenden Optionen, um das System an die vorhandene APM-Funkionalität anzupassen:

```
#options     APM_NO_IDLE              # Don't call BIOS CPU idle function
```

Die CPU-Idle-Funktion im BIOS sollte man nicht aufrufen, da manche Systeme dadurch bei bestimmten Gerätezugriffen (Soundkarte, Floppy) dabei hängenbleiben.

```
#options     APM_V10_ONLY             # Use only the APM 1.0 calls
```

Beschränkt das APM-Interface auf Version 1.0, da manche Systeme Version 1.1 nur halbherzig oder sehr eigenwillig implementieren.

```
options      APM_NO_V12
```

Wie *APM_V10_ONLY*, nur für Version 1.2.

```
options      APM_NO_STANDBY
```

Schaltet den Übergang zum Standby-Zustand ab.

```
#options     APM_NO_POWEROFF          # Don't power off on halt(8)
```

Wenn das System angehalten wird, soll nicht versucht werden, die Maschine abzuschalten. Ist *APM_NO_POWEROFF* nicht eingeschalten ist, kann ein Rechner mit ATX-Netzteil durch die Befehle *halt -p* und *poweroff* vollständig abgeschaltet werden.

```
options      APM_FORCE_64K_SEGMENTS
```

Erzwingt die Länge von 64 Kbyte für das Daten- und Codesegment des APM-BIOS.

```
options      APM_ALLOW_BOGUS_SEGMENTS
```

Erlaubt die Verwendung von Datensegmenten in ungewöhnlichen Regionen.

```
options      APMDEBUG
```

Schaltet den Debug-Modus ein, APM-Events werden damit auf der Konsole mitprotokolliert.

```
options      APMCALLDEBUG
```

Schaltet einen weiteren Debug-Modus ein, jeder Aufruf einer APM-Routine im BIOS wird mitprotokolliert. Diese Option erzeugt sehr viel Debug-Output auf der Konsole und ist deswegen nicht zu empfehlen.

```
#options     APM_POWER_PRINT                # Print stats on the console
```

Der Zustand der Stromversorgung (Ladezustand, Verbindung zum Stromnetz) wird auf der Konsole mitprotokolliert. Jedes Ereignis *APM_POWER_CHANGE* erzeugt einen Eintrag.

```
#options     APM_DISABLE_INTERRUPTS=0       # Don't disable interrupts
```

Wenn der Wert auf 0 gesetzt ist, wird der Kernel Interrupts während Aufrufen von APM-Funktionen nicht abschalten, da manche neueren Laptops sonst beim Aufwachen vom Suspend-Status hängenbleiben.

BIOSsiges

Normalerweise sollte ein gut erzogenes BIOS die Maschine durchgehend konfigurieren, Interrupts vergeben, I/O- und Speicherbereiche zuteilen – nur leider hinkt die Realität diesem Idealbild beizeiten hinterher. Daher gibt es die Option *PCIBIOS*, um Konfigurationsdaten aus dem BIOS zu extrahieren und die Maschine entsprechend zu konfigurieren.

```
#options     PCIBIOS                        # PCI BIOS support
```

Schaltet PCI-BIOS-Unterstützung ein.

```
#options     PCIBIOSVERBOSE                 # PCI BIOS verbose info
```

Bringt viel Information während der Konfigurationsphase.

```
#options     PCIBIOS_ADDR_FIXUP       # fixup PCI I/O addresses
#options     PCIBIOS_BUS_FIXUP        # fixup PCI bus numbering
#options     PCIBIOS_INTR_FIXUP       # fixup PCI interrupt routing
#options     PCIBIOS_IRQS_HINT=0x0a00 # PCI interrupts hint. IRQ 9 or 11
#options     PCIBIOS_INTR_GUESS       # see pcibios(4)
#options     PCIINTR_DEBUG            # super-verbose PCI interrupt fixup
```

Schalten nachträgliche Korrekturen der vom BIOS übergebenen Konfigurationen ein.

Sonstiges

```
options CPURESET_DELAY=2000
```

Definiert die Zeitdauer bei einem Reboot, bevor ein Hardwarereset durchgeführt wird, um es dem Benutzer zu ermöglichen, einen kurzen, vergänglichen Blick auf eine eventuelle Fehlermeldung zu erhaschen. Der Wert ist in Millisekunden, Standardeinstellung ist zwei Sekunden. Ein Wert von 0 schaltet die Pause ab.

```
options USER_LDT
```

Bindet Systemaufrufe zum Verändern der LDT (Local Descriptor Table) ein. Diese Option wird ausschließlich vom Windows-Emulator WINE verwendet.

Busadapter

Wie in Unterkapitel 4.1.5 beschrieben, sieht NetBSD einen Computer als ein Netz untereinander verbundener Bus-Systeme. Um das Entschlüsseln der Bootmeldungen ein wenig zu erleichtern, sind in Tabelle 4.1 die Gerätetreibernamen für die gebräuchlichsten Subsysteme zusammengefaßt.

Manche Bussysteme erlauben es, die angeschlossenen Geräte zu erkennen und handhaben auch die Vergabe von Ressourcen (I/O-Bereiche und Interrupts) selbständig, wie etwa der PCI-Bus. Daher erfolgen Gerätedefinitionen auf PCI immer in der Form

```
foo* at pci?
```

Andere, ältere Bussysteme wie etwa ISA haben keine Möglichkeit, herauszufinden – außer durch »Ausprobieren« –, welche Karten angeschlossen sind. Des weiteren ist die Ressourcenvergabe vollkommen in den Händen des Benutzers (meistens über Steckbrücken oder ein MS-DOS-Programm einzustellen). Daher sind für Geräte am ISA-Bus ein paar explizite Parameter notwendig:

```
foo* at isa? port 0x3e0 irq 9 iomem 0xcc000 iosiz 0x1000
```

Je nach Gerät sind nur manche dieser Angaben erforderlich. Bitte beachten Sie, daß die Angaben der Realität entsprechen müssen. Zwar kann die Suche nach einem falsch definierten Gerät durchaus Erfolg haben, funktionieren wird es dann allerdings oft nicht. Ein Beispiel dafür sind Meldungen der Form »ne0: device timeout«, die darauf hindeuten, daß eine Karte (in unserem Beispiel eine NE-2000-kompatible Netzwerkkarte) zwar mit der richtigen Port-Adresse eingebunden und dort auch gefunden wurde, jedoch die Angabe des IRQ falsch ist[1].

[1] Manche »moderne« NE-2000-kompatible Karten lassen sich nicht mehr durch Steckbrücken konfigurieren, sondern nur noch über ein DOS- oder Windows-Programm.

Name	Bus
isa	ISA-Bus
eisa	EISA-Bus
pci	PCI-Bus
pchb	PCI-Host Bridge
pceb	PCI-EISA Bridge. Verbindet PCI- und EISA-Bus.
pcib	PCI-ISA Bridge. Verbindet PCI- und ISA-Bus.
pcmb	PCI-MCA Bridge. Verbindet PCI und MCA.
ppb	PCI-PCI Bridge. Verbindet einen PCI- mit einem anderen PCI-Bus.
agp	AGP-Bus.
pcmcia	PCMCIA-Steckplatz.
mca	MCA-Bus.
pcic	PCMCIA-Controller.
isapnp	ISA-PNP-Bus.
cbb	Cardbus Bridge.

Tabelle 4.1: Bus-Treiber

Die Unterstützung des BIOS für PCMCIA ist bei manchen Motherboards leider nicht optimal, weshalb es manchmal zu IRQ- und Ressourcenkonflikten kommt. Die folgenden Optionen sollen helfen, auch nicht kooperierende Hardware zum Laufen zu bringen:

```
options PCIC_ISA_ALLOC_IOBASE=0x300
options PCIC_ISA_ALLOC_IOSIZE=0x0ff
```

Damit wird die I/O-Adresse und Größe für die Einblendung eines PCMCIA-Busses eingestellt.

```
options PCIC_ISA_INTR_ALLOC_MASK=mask
```

Stellt die für PCMCIA zugelassenen Interrupts ein, da es manchmal zu Konflikten mit anderen Geräten kommt oder manche Interruptleitungen gar nicht durchgeführt sind. Dieser Wert ist ein Bitfeld, *0x0001* steht für IRQ 0, *0x0002* für IRQ 1, bis zu *0x8000* für IRQ 15.

Harddisk

NetBSD unterstützt eine Vielzahl von IDE-, SCSI- und RAID-Festplatten-Controllern. Die meisten Anwendungen dafür sind Festplatten und CD-ROMs, es lassen sich aber durchaus auch Scanner oder Streamer an diesen Anschlüssen betreiben.

SCSI

Die meisten handelsüblichen SCSI-Hostadapter werden von NetBSD automatisch erkannt und eingebunden, die folgende Tabelle gibt einen Überblick über die bedienbaren Controller.

Device	Controller
adv	AdvanSys 1200[A,B], 9xx[U,UA]; AdvanSys APB-5140, 5142; AdvanSys 1200[A,B], 9xx[U,UA]
adw	AdvanSys 9x0UW, 3940UW
aha	Adaptec 1540, 1542; Adaptec AHA-1640; Buslogic BT-54x
ahb	Adaptec 1740, 1742
ahc	Auf den Chips Adaptec AIC77xx und AIC78xx basierende Hostadapter • Adaptec 294x, 394x • Adaptec 274x; • Adaptec ADP-1480 • Adaptec 284x
aic	Adaptec APA-1460; Adaptec AHA-1520B; Adaptec 1520, 1502
bha	BusLogic 9xx ; BusLogic 7xx; BusLogic 44x, 54x, 74x
dpt	DPT SmartCache/SmartRAID DPT EATA • DPT SmartCache III und IV • DPT SmartRAID III und IV
esp	NCR 53C90, 53C94, 53C96, Qlogic FAS216 und FAS408 basierend • Qlogic ESP406/FAS408 • Emulex ESP100, ESP100A, ESP200 and ESP406
iha*	Initio INIC-940 and INIC-950 basierend • INI-9090U, INI-9100U/UW • Iwill 2935UW • IOI Technology IOI-4203U

Tabelle 4.2 (Fortsetzung auf nächster Seite)

Device	Controller
iopsp	I2O SCSI-Host-Adapter
isp	Qlogic ISP [12]0x0 SCSI/FibreChannel • Qlogic ISP1000, 1020, 1040, 1240, 2100, 2200, 2300 • Performance Technology PTI SBS440, PTI SBS450, PTI SBS470 • Antares Microsystems P-0033
mly	Mylex AcceleRAID and eXtremeRAID • AcceleRAID 160, 170 und 352 • eXtremeRAID 2000 und 3000
nca	NCR-5380/NCR-53C400 basierend
osiop	Symbios/NCR 53C710 (nicht 53C700) basierend. Für Controller mit Symbios/NCR 53C8xx ist *siop* der passende Treiber.
pcscp	AMD 53c974 PCscsi-PCI basierend, etwa den Tekram DC-390 PCI. Für Tekram DC-390U/UW/F ist *siop* der passende Treiber. Für Tekram DC-395U/UW/F ist *trm* der passende Treiber.
sea	Seagate/Future Domain • ST01/02 • Future Domain TMC-885 und Future Domain TMC-950
siop	Symbios Logic/NCR 53x8xx • 53c810, 53c810a und 53c815 (Fast SCSI) • 53c820, 53c825 und 53c825a (Fast-Wide SCSI) • 53c860 (Ultra SCSI) • 53c875 und 53c875j (Ultra-Wide SCSI) • 53c876 (Dual Ultra-Wide SCSI) • 53c885 (Ultra-Wide SCSI und Ethernet) • 53c895 (Ultra2-Wide SCSI) • 53c896 (PCI 64bit, dual Ultra2-Wide SCSI) • 53c1010-33 (PCI 64bit, dual Ultra160 SCSI) • 53c1510d (PCI 64bit, dual Ultra2-wide SCSI)
trm	Tekram TRM-S1040-basierend • Tekram DC-315 und Tekram DC-395 Für Tekram DC-390 ist *pcscp* der passende Treiber. Für Tekram DC-310/U und DC-390U/UW/F ist *siop* der passende Treiber.

Tabelle 4.2 (Fortsetzung auf nächster Seite)

Device	Controller
uha	UltraStor 24f; UltraStor [13]4f
wds	WD7000 und TMC-7000, unter anderen: • WD7000-ASC Busmastering DMA-Controller • WD7000-ASE ASC mit Floppy und ESDI-Controller, für Apollo-Workstations • WD7000-FASST2 ASC mit neuer Firmware.

Tabelle 4.2: SCSI-Controller

Der GENERIC-Kernel enthält für SCSI-Geräte die folgenden Definitionen:

```
# SCSI devices
sd*     at scsibus? target ? lun ?    # SCSI disk drives
st*     at scsibus? target ? lun ?    # SCSI tape drives
cd*     at scsibus? target ? lun ?    # SCSI CD-ROM drives
ch*     at scsibus? target ? lun ?    # SCSI autochangers
ses*    at scsibus? target ? lun ?    # SCSI Enclosure Services devices
ss*     at scsibus? target ? lun ?    # SCSI scanners
uk*     at scsibus? target ? lun ?    # SCSI unknown
```

Wie zu sehen ist, sind die Gerätenummern alle durch Wildcards (das »*«) definiert – das hat einerseits den angenehmen Effekt, daß beliebig viele Geräte angeschlossen, erkannt und automatisch durchnumeriert werden. So wird der ersten SCSI-Disk damit das Device *sd0* zugewiesen, der zweiten SCSI-Disk *sd1* und so weiter. Allerdings hat dieses System auch den Nachteil, daß bei Ausfall einer Platte beziehungsweise dem Hinzufügen »mittendrin« alle nachfolgenden Gerätenummern auch verschoben werden.

Um das zu vermeiden, ist es meistens sinnvoll, die Geräte fest an Devicenummern zu binden. Dafür muß man für jedes SCSI-Gerät eine Zeile in die Konfigurationsdatei einfügen:

```
sd0     at scsibus? target 1 lun ?     # SCSI disk drives
sd1     at scsibus? target 4 lun ?     # SCSI disk drives
sd*     at scsibus? target ? lun ?     # SCSI disk drives
```

Damit wird der Platte mit ID 1 immer das Device *sd0*, ID 4 immer *sd1* zugeordnet. Der abschließende Eintrag mit Wildcard bewirkt, daß weitere Platten dann ab *sd2* eingeordnet werden. Ließe man den Wildcard-Eintrag weg, würden dagegen immer nur *sd0* und *sd1* eingebunden, selbst wenn noch zusätzliche Platten angeschlossen würden.

IDE

IDE-Geräte werden durch die Geräte *wdc* und *pciide* angesteuert. Man kann hier natürlich wieder den nicht verwendeten Treiber entfernen (modernere Maschinen mit PCI-Bus und On-Board-IDE verwenden meist *pciide*, ältere meistens *wdc*).

pciide unterstützt unter anderen die folgenden Controller:

- Acard ATP850 (Ultra/33) and ATP860 (Ultra/66) IDE-Controller.
- Acer Labs M5229 IDE Controller.
- Advanced Micro Devices AMD-756, 766 and 768 IDE-Controller.
- CMD Tech PCI0643, PCI0646, PCI0648 and PCI0649 IDE-Controller.
- Contaq Microsystems/Cypress CY82C693 IDE-Controller.
- HighPoint HPT366 Ultra/66 and HPT370 Ultra/100 IDE-Controller.
- Intel PIIX, PIIX3 and PIIX4 IDE-Controller.
- Intel 82801 (ICH/ICH0/ICH2/ICH4) IDE-Controller.
- OPTi 82c621 IDE-Controller.
- Promise PDC20246 (Ultra/33), PDC20262 (Ultra/66), PDC20265/PDC20267 (Ultra100), PDC20268 (Ultra/100TX2 und Ultra/100TX2v2) und Ultra/133 PCI IDE-Controller.
- Silicon Integrated System 5597/5598 IDE-Controller.
- VIA Technologies VT82C586, VT82C586A, VT82C596A, VT82C686A und
- VT8233A IDE-Controller.

Die folgende Zeile bindet den *pciide*-Treiber in den Kernel ein:

```
# IDE and related devices
pciide* at pci? dev ? function ? flags 0x0000
```

Über den Parameter *flags* können noch spezifische Eigenschaften an- und ausgeschaltet werden: Wird er auf 0x0001 gesetzt, verwendet der Treiber immer DMA, auch wenn der IDE-Controller unbekannt ist und kein spezifischer Code zur Ansteuerung vorhanden ist. Das kann funktionieren, aber auch nicht.

Im Gegenzug dazu zwingt ein flags-Wert von 0x0002 den Treiber dazu, DMA abzuschalten, auch wenn der erkannte Chipsatz eigentlich DMA unterstützte.

Der *wdc*-Treiber wird durch die folgenden Einträge definiert. Wie man sieht, sind Attachments an verschiedenen Bussystemen möglich:

```
wdc*      at isapnp?
wdc*      at pcmcia? function ?
wdc0      at isa? port 0x1f0 irq 14 flags 0x00
wdc1      at isa? port 0x170 irq 15 flags 0x00
```

Hier kann durch Setzen des Parameters *flags* auf 0x0001 der Treiber zur Verwendung von 32-Bit-Datentransfers bewegt werden. Das ist sinnvoll, falls der IDE-Controller zwar 32-Bit-I/O unterstützt, diese Fähigkeit aber vom Treiber nicht erkannt wird. Ein Wert von 0x0002 unterbindet die Suche nach einer zweiten Platte.

Die Platten per se werden nun durch

```
wd*     at wdc?    channel ? drive ? flags 0x0000.
wd*     at pciide? channel ? drive ? flags 0x0000
```

definiert. Hier kann der Parameter *flags* den zu verwendenden PIO- und DMA-Modus der angeschlossenen Platte bestimmen: das untere Nibble definiert den PIO-Modus, das nächstfolgende Nibble den DMA-Modus, und schließlich den Ultra-DMA-Modus – für DMA und UDMA wird der Wert 0xf verwendet, um diesen Modus generell auszuschalten. Für jeden dieser Werte bestimmen die drei unteren Bits den jeweiligen Modus, während das oberste Bit gesetzt werden muß, um diese Einstellung auch tatsächlich zu verwenden. Also würde zum Beispiel ein *flag*-Parameter von *0xaf0* bedeuten: Ultra-DMA-Modus 2, DMA-Modus deaktivieren, PIO-Modus nach den Vorgaben der Platte einstellen.

RAID

Auch RAID-Controller werden meist automatisch richtig erkannt. Die im BIOS des RAID-Controllers definierten Platten erscheinen dann als *ld* (Logical Disk) anstelle von *wd* oder *sd*.

Serielle Schnittstellen

Serielle Schnittstellen sind wohl eine der ältesten I/O-Schnittstellentypen, die heute noch in Verwendung sind. Neben den standardmäßig in jedem PC vorhandenen COM-Schnittstellen unterstützt NetBSD auch eine Vielzahl zusätzlicher Schnittstellenkarten.

Die Parameter für den ersten und zweiten eingebauten seriellen Port (*com1* und *com2* am ISA-Bus) sind faktisch standardisiert – jede weitere Schnittstelle kann jedoch an beliebige I/O und IRQs[1] gebunden werden.

```
#options    COM_HAYESP              # adds Hayes ESP serial board support
```

Diese Option schaltet im COM-Treiber Unterstützung für eine Hayes-ESP-Schnittstellenkarte ein.

[1] Allerdings funktioniert Interrupt-Sharing, außer bei speziellen Schnittstellenkarten, meist nicht – jeder serielle Port braucht einen eigenen zugewiesenen IRQ. Dies ist eine Einschränkung der Hardware.

Device	Controller
aac	Adaptec AAC-Familie
cac	Compaq EISA Array-Controller • Compaq Integrated Array • Compaq IAES • Compaq IDA • Compaq IDA-2 • Compaq RAID LC2 • Compaq Smart Array • Compaq SMART • Compaq SMART-2
icp	ICP-Vortex GDT & Intel RAID
mlx	Mylex DAC960 & DEC SWXCR family • DEC/Compaq SWXCR • Mylex AcceleRAID 150, 200, 250 • Mylex DAC1164P (eXtremeRAID 1100) • Mylex DAC960
twe	3ware Escalade IDE-RAID controllers

Tabelle 4.3: RAID-Controller

Parallele Schnittstellen

Die parallele Schnittstelle zum Betrieb eines Druckers wird durch den *lpt*-Treiber bedient.

```
# ISA parallel printer interfaces
lpt0        at isa? port 0x378 irq 7    # standard PC parallel ports

# PCI parallel printer interfaces
lpt*        at puc? port ?              # ports on "universal" comm boards
```

Es sind sowohl Attachments auf den standardisierten Ports auf dem ISA-Bus, also auch an PCI-Schnittstellen definiert. Die IRQ-Definition kann bei ISA auch weggelassen werden; damit wird der Treiber die Schnittstelle dann im Polling-Modus ansteuern.

Device	Karte
addcom	Addonics FlexPort 8S
ast	AST 4-port Serielle Karten
boca	Multiserielle Karten von BOCA Research • BB1004 (4 Ports) • BB1008 (8 Ports) • BB2016 (16 Ports)
com	Serielle Standardports im PC (NS8250-, NS16450- und NS16550 basierend)
cy	Serielle Karten von Cyclades • Cyclades Cyclom-4Y • Cyclom-8Y • Cyclom-16Y, • Cyclom-32Y
cz	Cyclades-Z multi-port serial boards • Cyclades-8Zo rev. 1, rev. 2 • Cyclades-Ze
moxa	MOXA C168H Serielle Karte (experimentell)
puc	PCI »universal« communications card • Dolphin Peripherals 4014 (dual parallel) • Dolphin Peripherals 4035 (dual serial) • NetMos NM9835 (dual parallel and single serial) • SIIG Cyber 2P1S PCI (dual parallel and single serial) • SIIG Cyber 2S1P PCI (dual serial and single parallel) • SIIG Cyber 4S PCI (quad serial) • SIIG Cyber I/O PCI (single serial and single parallel) • SIIG Cyber Parallel Dual PCI (dual parallel) • SIIG Cyber Parallel PCI (single parallel) • SIIG Cyber Serial Dual PCI (dual serial) • SIIG Cyber Serial PCI (single serial) • Vscom PCI-800 (8 port serial)
rtfps	RT 4-Port serielle Karten
tcom	Byte Runner Technologies • TC-400 (4 Ports) • TC-800 (8 Ports)

Tabelle 4.4: Serielle Schnittstellen

Netzwerk

BSD hat eine reiche Tradition an Netzwerkprotokollen, weshalb der Netzwerk-Teil der Kernelkonfiguration besonders üppig ausgefallen ist.

Grundlegende Optionen

```
options         INET                    # IP + ICMP + TCP + UDP
```

Schaltet den TCP/IP-Protokollstack zu. Diese Option ist Pflicht, denn viele Programme verwenden intern TCP/IP zur Kommunikation, auch wenn der Rechner gar keine Netzwerkkarte oder Verbindung zur Außenwelt besitzt.

```
options         INET6                   # IPV6
```

Im Gegensatz dazu kann man die Unterstützung für das neue IP-Protokoll IPv6 durchaus ein- oder ausschalten. Will man IPv6 nicht einsetzen, sollte man *INET6* deaktivieren, wodurch sich die Kernelgröße merklich reduziert.

```
#options    IPSEC           # IP security
#options    IPSEC_ESP       # IP security (encryption part; define w/IPSEC)
#options    IPSEC_DEBUG     # debug for IP security
```

Diese Optionen schalten die IPSec-Zusatzfunktionen ein. Um ein VPN mittels IPSec aufzubauen benötigt man *IPSEC* (enthält Schlüsselverwaltung, Policyverwaltung, AH und IPComp) und den Verschlüsselungsteil *IPSEC_ESP*[1]. IPSec kann sowohl mit IPv6 (*INET6*) als auch mit IPv4 (*INET*) verwendet werden.

```
pseudo-device   loop                    # network loopback
```

Durch *loop* wird das Loopback-Interface (IP-Adresse 127.0.0.1) realisiert. Es ist für den Betrieb des Rechners unabdingbar.

```
#options        GATEWAY                 # packet forwarding
```

Besitzt ein Rechner mehrere Netzwerkkarten und soll dazwischen Routingfunktionen übernehmen, muß man *GATEWAY* einschalten. Dadurch wird auch automatisch *IPFORWARDING* eingeschaltet und *NMBCLUSTERS* (siehe dort) wird auf einen höheren Wert gesetzt.

---

[1] *IPSEC_ESP* wurde ursprünglich ausgegliedert, um nicht in Konflikt mit – mittlerweile aufgehobenen – US-Ausfuhrbestimmungen zu geraten.

```
options IPFORWARDING=0
```

Soll ein Rechner zwischen mehreren Netzwerkkarten routen, muß dieser Schalter auf 1 gesetzt werden (das geschieht durch *GATEWAY* automatisch). Diese Option kann auch zur Laufzeit mittels *sysctl(8)* über die Variable *net.inet.ip.forwarding* gesteuert werden.

```
options IPFORWSRCRT=1
```

Erlaubt (Defaulteinstellung) oder verbietet die Weiterleitung von »source routed« Datenpaketen. Diese Option kann auch zur Laufzeit mittels *sysctl(8)* über die Variable *net.inet.ip.forwsrcrt* verändert werden. Sicherheitsrelevante Maschinen (Firewalls) sollten diese Option auf jeden Fall auf 0 setzen.

```
options NMBCLUSTERS=1024
```

Der Kernel verwaltet über das Netz transportierte Daten in sogenannten »mbuf-Clustern«. Bei Maschinen, die besonders hohem Netzwerkverkehr ausgesetzt sind, wie etwa Router oder Firewalls, kann es zu einer Verknappung dieser Cluster kommen. Der Kernel beschwert sich dann mit

```
mclpool limit reached: increase NMBCLUSTERS
```

und man sollte mit obiger Option die Anzahl der Cluster erhöhen, um den reibungslosen Betrieb zu gewährleisten.

```
#options MROUTING
```

Bindet Unterstützung für Multicast-Routing ein. Nicht sehr gebräuchlich.

Sonstige Protokolle

Neben IP und IPv6 unterstützt NetBSD noch einige eher selten verwendete Protokolle, die man guten Gewissens abschalten kann.

```
#options     MROUTING          # IP multicast routing
options      NS                # XNS
#options     NSIP              # XNS tunneling over IP
options      ISO,TPIP          # OSI
#options     EON               # OSI tunneling over IP
options      CCITT,LLC,HDLC    # X.25
options      NETATALK          # AppleTalk networking protocols
```

Außer *Netatalk* wird man kaum eines dieser Protokolle jemals zu Gesicht bekommen. *Netatalk* schaltet Support für das Appletalk-Protokoll ein, zusätzlich zu dieser Option

wird noch ein Userland-Server benötigt, der aus dem Package-System installiert werden muß (Package »net/netatalk«).

PPP und SLIP

PPP und SLIP sind beides Protokolle, die zum Betrieb von Punkt-zu-Punkt-Verbindungen erfunden wurden. SLIP (Serial Line IP) ist ein sehr einfaches Protokoll, das aber heutzutage kaum mehr eingesetzt wird, da es von PPP (Point to Point Protocol) abgelöst wurde, das deutlich leistungsfähiger (und einfacher zu konfigurieren) ist.

```
pseudo-device   ppp             2       # Point-to-Point Protocol
pseudo-device   pppoe                   # PPP over Ethernet (RFC 2516)
pseudo-device   sl              2       # Serial Line IP
```

Man kann diese Einträge problemlos entfernen, wenn man nicht vorhat, PPP oder SLIP einzusetzen. Die Zahlen bestimmen die Anzahl der möglichen Pseudointerfaces (mit der obigen Definition können dann beispielsweise *ppp0* und *ppp1* verwendet werden). Das *pppoe*-Device ist ein wenig neuer: Es können beliebig viele Instanzen zur Laufzeit generiert werden, denn es handelt sich um ein sogenanntes »Cloning Interface«, neue Instanzen müssen vor der Verwendung durch *ifconfig pppoe0 create* erzeugt werden.

Tunnelprotokolle

Die folgenden Pseudo-Devices werden zur Implementation von diversen Tunnelprotokollen verwendet. *tun* ist ein generischer Tunnelendpunkt, der verwendet werden kann, um Protokolle in Userland-Programmen zu realisieren. *GRE* ist eine generische Verpakkung (Generic Routing Encapsulation), *gif* verpackt IP/IPv6 in IP/IPv6. Zu näherer Information bitte die zugehörigen Manpages befragen.

```
pseudo-device   tun             2   # network tunneling over tty
pseudo-device   gre             2   # generic L3 over IP tunnel
pseudo-device   gif             4   # IPv[46] over IPv[46] tunnel (RFC1933)
#pseudo-device  faith           1   # IPv[46] tcp relay translation i/f
#pseudo-device  stf             1   # 6to4 IPv6 over IPv4 encapsulation
pseudo-device   vlan                # IEEE 802.1q encapsulation
```

Sonstiges

```
pseudo-device       bpfilter 8      # Berkeley packet filter
options             PFIL_HOOKS      # pfil(9) packet filter hooks
```

Mit dem Berkeley Packet Filter kann man den Datenstrom auf einem Netzwerkinterface mithören, zum Beispiel mit *tcpdump(8)*.

```
#options        TCP_COMPAT_42    # 4.2BSD TCP/IP bug compat. Not recommended.
```

Schaltet den TCP-Kompatibilitätsmodus mit BSD 4.2 ein, womit ein paar Fehler in der 4.2-TCP-Implementierung nachgebildet werden. Diese Option sollte man generell ausgeschaltet lassen.

IP-Filter

```
pseudo-device       ipfilter        # IP filter (firewall) and NAT
```

NetBSD kann auch Firewall- oder Packetfilter-Funktionen übernehmen. Dafür ist das IP-Filter-Subsystem zuständig (analog zu *IPChains* in Linux oder *ipfw* in FreeBSDs). Durch die Einbindung des Pseudo-Devices *ipfilter* wird der Paketfilter-Code in den Kernel eingebunden.

```
options         IPFILTER_LOG            # ipmon(8) log support
```

Um zu sehen, was hinter dem Vorhang von *ipfilter* abläuft, gibt es exzessive Logging-möglichkeiten – allerdings nur, wenn *IPFILTER_LOG* auch eingeschaltet ist. Sollten Sie sich einmal wundern, warum Ihre *ipfilter*-Regeln keine Log-Einträge produzieren, kann es am Fehlen dieser Option liegen.

```
options IPFILTER_DEFAULT_BLOCK
```

Defaultmäßig werden ungefilterte Daten immer durchgelassen. Bei der Implementation einer Firewall ist es jedoch besser, eine restriktive Politik anzuwenden und ungefilterte Pakete grundsätzlich zu blockieren.

Ethernetkarten

Im Unterschied zu Solaris oder Linux werden bei BSD die Ethernetschnittstellen durch ihr Treiberkürzel beschrieben und nicht als generisches *eth0*. Es wird eine Vielzahl von Geräten unterstützt, die folgenden Tabellen geben eine kurze Übersicht:

Device	Netzwerkkarte
ai	AT&T StarLAN
	• AT&T StarLAN 10 und AT&T StarLan Fiber
ate	Fujitsu MB86960A/MB86965A-basierend
	• Allied-Telesis AT1700T/AT1700BT/AT1700FT/AT1700AT
	• Allied-Telesus AT1720T/AT1720BT/AT1720FT/AT1720AT
	• Allied-Telesis RE2000

Tabelle 4.5 (Fortsetzung auf nächster Seite)

Device	Netzwerkkarte
cs	Cirrus Logic Crystal CS8900-basierende Karten • IBM Etherjet
ec	3Com 3c503 (Etherlink II)
eg	3Com 3c505 (Etherlink Plus)
el	3Com 3c501 (Etherlink)
elmc	3Com 3c523 (EtherLink/MC)
ef	3Com 3C507 (Etherlink II)
en	ENI/Adaptec ATM • Efficient Networks, Inc. ENI-155 • Adaptec 5900 und 5905
ep	3Com 3c509, »Vortex«, »Boomerang« basierend: • 3Com 3c509 (Etherlink III), 3c509B, 3c529, 3c579, 3c562, 3c589 • 3Com 3c589, 3c562, 3c529 • 3Com 3c515, 3c905, 3c574 • 3Com 3c592, 3c579, 3c590, 3x595, 3c900-TPO, 3c900-COMBO Für 3c90x, 3c90xB und 3c575 (CardBus) ist ex der passende Treiber.
epic	SMC SMC83C170 (EPIC/100) basierend
esh	Essential Communications RoadRunner-basierende HIPPI
ex	3Com Fast EtherLink XL • 3c450 • 3c555, 3c556 • 3c575-TX/B-TX/CT • 3c900-TPO/COMBO • 3c900B-TPC/TPO/COMBO • 3c905-T4/TX • 3c905B-COMBO/FX/T4/TX • 3c980 Server Adapter • 3c980C-TXM Alle neueren Etherlink XL Adapter unterstützen TCP/UDP Prüfsummenberechnung in Hardware.

Tabelle 4.5 (Fortsetzung auf nächster Seite)

Device	Netzwerkkarte
fea, fpa, fta	DEC FDDI • DEFPA, DEFEA und DEFTA
fmv	Fujitsu MB8696xA-basierend • Fujitsu FMV-181 und Fujitsu FMV-182
fxp	Intel i82557-, i82558- und i82559-basierend • Intel EtherExpress Pro 10+ • Intel EtherExpress Pro 100B • Intel EtherExpress Pro 100+ • Intel PRO/100 S
gsip	National Semiconductor NS83820 Gigabit Ethernet • NetGear GA-622 • Asante FriendlyNet GigaNIX • D-Link DGE-500T • SMC 9452TX, 9462TX Der 83820 unterstützt VLAN-Funktionalität in Hardware, der DP83820 unterstützt TCP/UDP-Prüfsummenberechnung in Hardware.
ix	Intel Etherexpress/16
iy	Intel EtherExpress PRO 10
lc	DEC EtherWORKS III (LEMAC) • DE203, DE204, DE205
le	AMD 7990-, 79C90-, 79C960-, 79C970-basierend • BICC Isolan • Novell NE2100 • Digital DEPCA • DEC DE422 • SKNET Personal MC2 • SKNET MC2+
lmc	DEC DECchip 21140-basierend • Lan Media Corp LMC1000, LMC1200, LMC5200, LMC5245
ne	NE2000 und Kompatible

Tabelle 4.5 (Fortsetzung auf nächster Seite)

Device	Netzwerkkarte
pcn	AMD PCnet-PCI-basierend • Am79c970, Am79c970A, Am79c971, Am79c972 • Am79c973/Am79c975 • Allied Telesyn AT-2700TX
rtk	Realtek 8129/8139-basierend
sf	Adaptec AIC-6915-basierend • ANA-62011, ANA-62022, ANA-62044 , ANA-62020 und ANA-69011
sip	SiS 900/DP83815 Ethernet
sm	SMC91C9x Ethernet
ste	Sundance ST-201 Ethernet • D-Link DFE-550TX
stge	Sundance/Tamarack TC9021 Gigabit • D-Link DGE-550T • Antares Microsystems Gigabit Ethernet TC9021 unterstützt TCP/UDP-Prüfsummenberechnung in Hardware.
ti	Alteon ACEnic (Tigon I und Tigon II) gigabit Ethernet • Alteon AceNIC V gigabit • 3Com 3c985-SX gigabit • Netgear GA620 • Digital EtherWORKS 1000SX (DEGPA) Algeon Tigon und Tigon II unterstützen TCP/UDP-Prüfsummenberechnung in Hardware.
tl	Texas Instruments ThunderLAN-basierend • Compaq Netelligent • Compaq NetFlex 3/P • Compaq Deskpro 4000 5233MMX Ethernet • TI TravelMate 5000 Serie Laptop-Docking-Station TI0
tlp	Auf Digital Equipment 21x4x-basierende Karten, DEC DE-425, DLINK DFE-570TX, KNE110TX

Tabelle 4.5 (Fortsetzung auf nächster Seite)

Device	Netzwerkkarte
tr	IBM/3COM TROPIC Token-Ring • IBM Token-Ring Network PC Adapter(/A) • IBM Token-Ring Network PC Adapter II • IBM Token-Ring Network 16/4 Adapter(/A) • IBM Token-Ring 16/4 Credit-Card-Adapter • IBM Token Ring Auto 16/4 Credit-Card-Adapter • IBM Turbo 16/4 Token-Ring-PC-Card • IBM 16/4 ISA-Adapter • IBM Auto 16/4 Token-Ring ISA-Adapter • IBM Token Ring 16/4 Credit-Card-Adapter • IBM Token Ring Auto 16/4 Credit-Card-Adapter • IBM Turbo 16/4 Token-Ring-PC-Card • 3COM 3C619 TokenLink • 3COM 3C319 TokenLink Velocity • 3COM 3C389 TokenLink Velocity PC Card
vr	VIA VT3043 (Rhine) and VT86C100A (Rhine-II)-basierend • D-Link DFE530TX
we	Western Digital/SMC WD80x3-basierend • Western Digital/SMC WD80x3 • SMC Elite Ultra • SMC EtherEZ
wm	Intel 82543/82544 gigabit
xi	Xircom CreditCard Ethernet • Xircom CreditCard Ethernet II • Xircom CreditCard 10/100 Ethernet • Xircom RealPort 10/100 Ethernet + Modem (nur Ethernet-Funktion) • Intel EtherExpress Pro/100 • Compaq Netelligent 10/100

Tabelle 4.5: Gerätetreiber für Ethernet

Device	Wireless-LAN-Karte
an	Aironet PC4500/PC4800 (802.11) • Aironet 4500 • Aironet 4800 • Cisco Aironet 340 • Cisco Aironet 350
awi	AMD PCnetMobile IEEE 802.11-basierend • BayStack 650 und BayStack 660 • Icom SL-200 • Melco WLI-PCM • NEL SSMagic • Netwave AirSurfer Plus und Netwave AirSurfer Pro • Nokia C020 WLAN • Farallon SkyLINE
cnw	Netwave AirSurfer • Xircom CreditCard Netwave • NetWave AirSurfer Für Netwave AirSurfer »Plus« ist *awi* der passende Treiber. Dieser Treiber benötigt etwa 35 Kbyte I/O-Bereich, möglicherweise muß man dazu den für PCMCIA verfügbaren Platz vergrößern (siehe pcmcia).
ray	Raytheon Raylink (802.11) • WebGear Aviator 2.4 • WebGear Aviator PRO • Raytheon Raylink WLAN
wi	WaveLAN/IEEE und PRISM-II 802.11 • Lucent Wavelan/IEEE 2.0Mb Bronze/Silver • Lucent WaveLAN/IEEE Turbo/Turbo 11Mbps • Melco AIR CONNECT WLI-PCM-L11, WLI-PCM-L11G • Melco AIR CONNECT WLI-CF-S11G • Compaq WL100 11Mbps Wireless • Corega Wireless LAN PCC-11, PCCA_11, PCCB_11 • DEC/Cabletron RoamAbout 802.11 DS High Rate • D-Link DWL-650 11Mbps WLAN Card • ELECOM Air@Hawk LD-WL11 • ELSA AirLancer MC-11 • Ericsson Wireless LAN

Tabelle 4.6 (Fortsetzung auf nächster Seite)

Device	Wireless-LAN-Karte
wi (2. Teil)	• Farallon Skyline 11Mbps Wireless • Intel PRO/Wireless 2011 LAN PC Card • ICOM SL-1100 • IO-DATA WN-B11/PCM • Intersil PRISM Mini-PCI • Linksys Group, Inc. Instant Wireless Network PC Card (Prism2) • NCR WaveLAN/IEEE • NEC Wireless Card CMZ-RT-WP, PK-WL001, PC-WL/11C • PLANEX GeoWave/GW-NS110 • Symbol Spectrum24 Wireless LAN • TDK LAK-CD011WL Karten, die auf dem PRISM-I-Chipset aufbauen, werden vom *awi*-Treiber unterstützt.

Tabelle 4.6: Gerätetreiber 802.11 für Wireless LAN

Device	PHY-Treiber
acphy	DAltima AC101 und AMD Am79c874 PHYs
amhphy	AMD 79c901 Ethernet PHYs
bmtphy	Broadcom BCM5201 und BCM5202 PHYs
brgphy	Broadcom BCM5400-family PHYs
dmphy	Davicom DM9101 PHYs
exphy	3Com internal PHYs
glxtphy	Level One LXT-1000 PHYs
gphyter	NS83861 Gig-E PHY
icsphy	Integrated Circuit Systems ICS189x
inphy	Intel 82555 PHYs
iophy	Intel 82553 PHYs
lxtphy	Level One LXT-970 PHYs
makphy	Marvell Semiconductor 88E1000 PHYs
nsphy	NS83840 PHYs
nsphyter	NS83843 PHYs

Tabelle 4.7: (Fortsetzung auf nächster Seite)

Device	PHY-Treiber
pnaphy	Generic HomePNA PHYs
qsphy	Quality Semiconductor QS6612 PHYs
sqphy	Seeq 80220/80221/80223 PHYs
tlphy*	ThunderLAN PHYs
tqphy	TDK Semiconductor PHYs
ukphy	Generic unknown PHYs
urlphy	Realtek RTL8150L internal PHYs

Tabelle 4.7: Ethernet-PHY

Neuere 100-Mbit-Ethernetkarten benötigen außerdem noch einen *PHY*-Treiber. Ein *PHY* ist ein unabhängiger Baustein, der die Umsetzung auf das physikalische Medium übernimmt, zum Beispiel die automatische Erkennung der Geschwindigkeit, Full- oder Halbduplex et cetera.

Sie können außer den von Ihren Netzwerkkarten verwendeten PHYs alle ausschalten.

Konsole

NetBSD kommt mit zwei unterschiedlichen Konsolen-Treibern, nämlich dem alten *pccons* und dem neueren *wscons*. *pccons* hat den Vorteil, sehr klein und kompakt zu sein. *wscons* im Gegenzug bietet mehr Features. *wscons* wurde entwickelt, um ein einziges für alle Plattformen verwendbares Konsolen-Interface zu schaffen, anstatt auf jeder Maschinenarchitektur einen eigenen Treiber pflegen zu müssen.

Im allgemeinen sollte man sich für *wscons* entscheiden, da *pccons* nicht weiterentwickelt wird (allerdings ist *pccons* deutlich kleiner).

```
# builtin terminal emulations
#options      WSEMUL_SUN              # sun terminal emulation
```

Emulation der Sun-Konsole einbinden. Auf i386-Maschinen normalerweise nicht nötig, denn wer hat schon eine Sun-Tastatur mit allen ihren Funktionstasten.

```
options       WSEMUL_VT100            # VT100 / VT220 emulation
```

Emulation von DEC-VT100-Terminals. Diese Option schaltet *wscons* in einen Modus, der mit VT100-Terminals weitestgehend kompatibel ist (mit einigen VT220- und VT320-Erweiterungen).

```
options       WS_KERNEL_FG=WSCOL_GREEN
#options      WS_KERNEL_BG=WSCOL_BLACK
```

```
#options        WS_KERNEL_COLATTR=WSATTR_XXX
#options        WS_KERNEL_MONOATTR=WSATTR_XXX
```

Diese Optionen erlauben es, Ausgaben des Kernels (zum Beispiel auch die Bootmeldungen) in unterschiedlicher Farbe und Attributen zu normalen Ausgaben zu gestalten. Die Farb- und Attributnamen befinden sich in der Datei »/usr/include/dev/wscons/wsdisplayvar.h«.

```
# compatibility to other console drivers
options         WSDISPLAY_COMPAT_PCVT         # emulate some ioctls
```

Schaltet eine beschränkte Emulation des alten »pcvt«-Konsolentreibers ein, um X11 zum Laufen zu bringen.

```
options         WSDISPLAY_COMPAT_SYSCONS      # emulate some ioctls
```

Schaltet eine beschränkte Emulation des »syscons«-Konsolentreibers von FreeBSD ein, um X11 zufriedenzustellen. Das ist für Applikationen sinnvoll, die in der FreeBSD-Emulationsumgebung laufen.

```
options         WSDISPLAY_COMPAT_USL          # VT handling
```

Unterstützung für das von den meisten PC-Unix-Versionen verwendeten Protokoll, um zwischen virtuellen Bildschirmen hin- und herzuschalten.

```
options         WSDISPLAY_COMPAT_RAWKBD       # can get raw scancodes
```

Aktiviert die Möglichkeit, XT-Tastaturscancodes abzufragen. Dieses Feature wird vom i386 X-Server verwendet.

```
# see dev/pckbc/wskbdmap_mfii.c for implemented layouts
#options       PCKBD_LAYOUT="(KB_DE | KB_NODEAD)"
```

Erlaubt die Anpassung der Konsole auf andere Tastaturlayouts, zum Beispiel die deutsche QWERTZ-Anordnung. Die Tastatur kann aber auch jederzeit durch *wsconsctl(8)* angepaßt werden.

```
# allocate a number of virtual screens at autoconfiguration time
#options        WSDISPLAY_DEFAULTSCREENS=4
```

Allokiert die angegebene Anzahl an umschaltbaren Konsolen zum Bootzeitpunkt. Da man aber auch später ohne Probleme mittels *wsconscfg(8)* zusätzliche Konsolen aktivie-

ren kann und dies sogar in der Standard-Bootprozedur schon vorgesehen ist, muß man diese Option nicht spezifizieren.

```
# use a large software cursor that doesn't blink
options         PCDISPLAY_SOFTCURSOR
```

Schaltet einen großen, nicht blinkenden Blockcursor ein. Diese Option ist wohl Geschmackssache, manche finden diesen Blockcursor potthäßlich und schalten den Softcursor in jeder Kernel-Konfigurationen sofort aus.

```
# modify the screen type of the console; defaults to "80x25"
#options        VGA_CONSOLE_SCREENTYPE="\"80x24\""
```

Stellt die Größe der Standardkonsole ein.

Die folgenden Optionen sind notwendig, um *wscons* an die vorhandene Hardware zu binden; wie immer kann man unbenutzte Geräte (zum Beispiel *pcdisplay*) auskommentieren:

```
# wscons
pckbc0          at isa?                         # pc keyboard controller
pckbd*          at pckbc?                       # PC keyboard
pms*            at pckbc?                       # PS/2 mouse for wsmouse

vga0            at isa?
vga*            at pci? dev ? function ?
pcdisplay0      at isa?                         # CGA, MDA, EGA, HGA
wsdisplay*      at vga? console ?
wsdisplay*      at pcdisplay? console ?
wskbd*          at pckbd? console ?
wsmouse*        at pms? mux 0

# wscons pseudo-devices
pseudo-device   wsmux                           # mouse & keyboard multiplexor
```

Das *wsmux*-Device verbindet mehrere Eingabeströme zu einem, um beispielsweise mehrere Tastaturen oder Mäuse anschließen zu können oder um synthetische Eingaben in die Konsole zu injiziieren.

```
pseudo-device   wsfont
```

Dieses Pseudodevice erlaubt das Nachladen benutzerdefinierter Fonts in die Konsole. Der Vollständigkeit halber hier die Konfigurationseinträge, die für *pccons* benötigt werden:

```
# ISA console
#pc0        at isa? port 0x60 irq 1   # pccons generic PC console driver
# Keyboard layout configuration for pccons
#options    FRENCH_KBD
#options    FINNISH_KBD
#options    GERMAN_KBD
#options    NORWEGIAN_KBD
# pccons-specific options:
#options    XSERVER_DDB       # PF12 gets you into DDB when X is running
#options    XSERVER           # X server support
```

Emulationen

NetBSD bietet einige Emulationsoptionen, mit denen ältere NetBSD-Versionen oder gar Programme für ganz andere Betriebssysteme ausgeführt werden können.

```
options    COMPAT_NOMID  # compatibility with 386BSD, BSDI, NetBSD 0.8,
options    COMPAT_09     # NetBSD 0.9,
options    COMPAT_10     # NetBSD 1.0,
options    COMPAT_11     # NetBSD 1.1,
options    COMPAT_12     # NetBSD 1.2,
options    COMPAT_13     # NetBSD 1.3,
options    COMPAT_14     # NetBSD 1.4,
options    COMPAT_15     # NetBSD 1.5,
```

Diese Optionen schalten die Binärkompatibilität zu älteren NetBSD-Systemen (von NetBSD 0.9 bis 1.5) ein. Ist das System frisch aufgesetzt und hat man keinerlei »Altlasten«, kann man die entsprechenden Funktionen abschalten.

```
options    COMPAT_43     # and 4.3BSD
```

Schaltet ein paar alte von BSD 4.3 verwendete Kernelfunktionen ein. Im Zweifelsfall sollte man sie eingeschaltet lassen, denn so manches Programm verläßt sich darauf (oder stammt noch aus jenen längst vergangenen Zeiten).

```
options    COMPAT_LINUX        # binary compatibility with Linux
options    COMPAT_FREEBSD      # binary compatibility with FreeBSD
```

Schaltet die Binärkompatibilität zu Linux oder FreeBSD ein. Dafür ist außerdem noch entsprechende Userland-Unterstützung notwendig (das sind Libraries und Tools des jeweiligen Betriebssystems). Am einfachsten ist eine funktionierende Linux- oder FreeBSD-Umgebung über das Package-System zu erhalten (Packages *emulators/suse_linux* und *emulators/freebsd_lib*).

Dateisysteme

NetBSD bietet eine Vielzahl an Dateisystemen. Das sind Treiber, die die Daten auf Festplatten organisieren und verwalten (Verwaltung von Verzeichnisstrukturen, Überprüfen von Zugriffsberechtigungen und ähnliches).

```
file-system     FFS         # UFS
```

Das Berkeley Fast File System (FFS) ist das Standard-Filesystem unter NetBSD. Man muß es unbedingt eingeschaltet lassen!

```
#options        FFS_EI      # FFS Endian Independent support
```

Schaltet den »Endian Independence«-Modus des FFS ein. Traditionellerweise hängen manche auf die Platte geschriebenen Datenstrukturen von der vom Prozessor vorgegebenen Byteordnung ab. Diese Option bindet Erweiterungen ein, die es erlauben, FFS-Dateisysteme zwischen Maschinen unterschiedlichen Byteordnung (etwa i386 und PowerPC) zu transportieren. Diese Option macht keinen Sinn, wenn man nicht vorhat, Platten physisch zwischen verschiedenen Plattformen zu transportieren.

```
options     SOFTDEP             # FFS soft updates support.
```

Bindet Softdep-Support ein. Softdep (oder Soft-Updates) ist eine Methode dafür, das Dateisystem permanent konsistent zu halten, indem Metainformation[1] erst später in einer bestimmten Reihenfolge auf die Platte geschrieben werden. Durch diese Verzögerung wird besonders beim Anlegen und Löschen von Dateien eine spürbare Beschleunigung erreicht. Softdep befindet sich noch in der Entwicklung und es gibt sehr unterschiedliche Meinungen dazu – bei manchen Anwendern funktioniert es problemlos auch unter großer Last, andere berichten von häufigen Abstürzen.

```
options         QUOTA               # UFS quotas
```

Quota ermöglichen die Beschränkung des für bestimmte User verfügbaren Plattenplatzes auf FFS-Dateisystemen. Wenn nicht benötigt, sollte man diese Option deaktivieren, da das Aktualisieren der Quota-Daten ein bißchen Performance schluckt.

```
file-system     NFS                 # Network File System client
```

Dieser Schalter bewirkt das Einbinden des Client-Teils des NFS (Network File Systems). Wenn man nicht vorhat, Dateien von anderen Maschinen über NFS zu importieren, kann man dieses Filesystem abschalten.

---

[1] Metainformation sind Einträge auf der Platte, die die Daten am Filesystem selber beschreiben, also Dateinamen, Verzeichnisse, Zugriffsrechte und ähnliches.

Quotas und ihre Verwaltung

Das Quota-System dient dazu, auf großen Datenträgern zu verhindern, daß User mehr als einen vorbestimmten Platz für ihre Daten verwenden können. Ohne Quotas können normale User den freien Platz bis zu hundert Prozent belegen. Erst dann meldet das System, daß keine Daten mehr untergebracht werden können. Bei der Angabe von hundert Prozent bleibt zwar immer noch ein Sicherheitspuffer, da mit *root*-Rechten mehr als hundert Prozent (je nach Größe des Datenträgers können es 107 Prozent und mehr sein) belegt werden können.

Der verschwenderische Umgang von Usern mit Speicherplatz wird verhindert, indem im Kernel Quotas eingeschaltet werden (im GENERIC-Kernel sind sie das) und anschließend in der Datei »/etc/rc.conf« der Steuerbefehl *quota=YES* definiert wird (das entsprechende Startskript ist »/etc/rc.d/quota«).

Nun muß in der Datei »/etc/fstab« definiert werden, auf welchen Datenträgern Quotas eingesetzt werden sollen. In der Optionensektion des FFS-Datenträgers muß definiert werden, ob User-Quotas, Gruppen-Quotas oder beides verwendet werden soll. Aus

```
/dev/wd2e /pub ffs rw 1 2
```

wird damit beispielsweise

```
/dev/wd2e /pub ffs rw,userquota,groupquota 1 2
```

Je nachdem, ob User-Quotas oder Gruppen-Quotas (oder beide) gewünscht sind muß jetzt noch auf dem Datenträger die Verwaltungsdatei mit *touch* angelegt werden. Im obigen Beispiel wären das:

```
touch /pub/quota.user
touch /pub/quota.group
```

Im laufenden Betrieb kann jetzt bereits mit dem Befehl

```
quotaon /pub
```

die Quota-Verwaltung eingeschaltet werden. Die frisch angelegten Dateien werden hierbei auch mit den entsprechenden Benutzerdaten gefüllt. Zum jetzigen Zeitpunkt sind aber noch keine Einschränkungen festgelegt. Sie müssen mit dem Programm *edquota* für jeden Benutzer beziehungsweise die Gruppe definiert werden:

```
edquota -u user1 user2 user3 ...
```

Für jeden der auf der Kommandozeile angegebenen Benutzer wird ein Fenster des Systemeditors geöffnet, in dem sich die Werte angeben lassen:

```
Quotas for user user1:
/pub: blocks in use: 746732, limits (soft = 0, hard = 0)
      inodes in use: 1331, limits (soft = 0, hard = 0)
```

Nun müssen die entsprechend maximal zur Verfügung gestellten Blöcke eingetragen und die geänderte Datei gespeichert werden. Mit dem voreingestellten Wert 0 sind keine Einschränkungen definiert. Das heißt, daß die Quota-Verwaltung gezielt durchgeführt werden kann und User, denen uneingeschränkter Zugriff gewährt wird, einfach belassen werden können. Auf dem privaten Home-Server sind Quotas sicher eine übertriebene Maßnahme, auf einem Firmenserver aber ein einfaches Mittel zur ausgewogenen Verteilung der Ressourcen.

Zur Quota-Verwaltung gehören die Programme *quota*, *quotaon*, *quotaoff*, *quotacheck*, *edquota*, *quotactl* und *repquota*. Weitere Informationen zu ihrer Bedienung geben die Manpages der Programme. Die Quota-Verwaltung ist nur durch den Administrator *root* möglich.

```
options         NFSSERVER           # Network File System server
```

Bindet den Serverteil von NFS ein. Im Gegensatz zu *file-system NFS* ist *NFSSERVER* nur notwendig, wenn Sie Daten von Ihrer Maschine per NFS exportieren möchten.

```
options         NFS_BOOT_DHCP,NFS_BOOT_BOOTPARAM
```

Diese Schalter erlauben es, ein System ohne Platte zu betreiben, indem die Konfigurations- und Bootinformation mittels DHCP oder BOOTPARAM von einem entfernten Server übermittelt wird. Wenn man nicht mit diskless Boot arbeiten möchte, kann man diese Optionen ausschalten.

```
file-system     LFS             # log-structured file system
```

Experimentell. Log-strukturiertes Dateisystem.

```
file-system     MFS             # memory file system
```

Dieses Dateisystem legt die Daten im (virtuellen) Speicher ab, es ist also eine RAM-Disk und als solche auch besonders schnell. Durch einen Reboot ist freilich alles verloren, daher sollte man das MFS auch nur für temporäre Daten verwenden, zum Beispiel als schnelles */tmp*.

Das LFS-Dateisystem

In NetBSD 1.6 liegt das Dateisystem LFS inzwischen in der Version 2 vor. Trotzdem gilt es immer noch als experimentell und sollte tatsächlich auch so verstanden werden. Die Unterstützung ist deshalb auch rudimentär. Man kann LFS-Partitionen bei der Installation von NetBSD nicht einrichten und muß nach der Installation mit *fdisk* ein vorhandenes FFS-Dateisystem nach LFS konvertieren. Sie können das ausprobieren, sollten aber den experimentellen Status dieses Dateisystems nicht vergessen!

Zuerst muß mit *fdisk* die entsprechende NetBSD-Partition (Slice) angelegt sein. Dann wird im Disklabel ein Buchstabe ganz regulär mit dem Dateisystem »4.2BSD« definiert. Um diese Partition, beispielsweise auf der Platte »sd1«, umzuwandeln, wird *disklabel* mit dem Parameter *-e* aufgerufen. Dazu darf diese Partition aber nicht gemountet sein und sollte in der »/etc/fstab« auch auskommentiert sein:

Im Disklabel-Editor werden nun mit *disklabel -e sd1* die Daten geändert. Das Disklabel sieht auf der Beispielfestplatte ursprünglich so aus (entscheidend sind die fett markierten Angeben). Es handelt sich nicht um die Boot-Platte, deshalb ist die Partition »a« nicht belegt:

```
8 partitions
#       size offset fstype [fsize bsize cpg]
b:  262593      63 swap                    # Cyl.    0-255
c: 8467137      63 unused      0     0     # Cyl.    0-8252
d: 8467137       0 unused      0     0     # Cyl.    0-8252
e: 5119740  262656 4.2BSD     1024  8192  16 # Cyl.  256-5245
f: 3084804 5382396 4.2BSD     1024  8192  16 # Cyl. 5246-8252
```

Die Daten der Partition »e« werden jetzt zur Vorbereitung auf das LFS-Dateisystem angepaßt:

```
e: 5119740  262656 4.4LFS     1024  8192   8
```

Anschließend wird das neue Disklabel gespeichert und der Editor verlassen. Jetzt muß mit *mkfs_lfs /dev/sd1e* die Partition noch formatiert werden, was beim LFS-Dateisystem enorm schnell geht. Ein Aufruf von *fsck_lfs /dev/sd1e* prüft, ob alle Vorgänge korrekt waren. Nun muß noch die »/etc/fstab« angepaßt werden:

```
/dev/sd1e /home lfs rw 1 2
```

Das war schon alles. Ab sofort kann die Partition verwendet werden, wie ein *mount /home* beweist. Daß es sich nicht um eine Partition im ffs-Format handelt, zeigen die Datei »ifile« und das Verzeichnis »lost+found« im Verzeichnis »/home«, auf das die Partition eingehängt wurde.

```
file-system   EXT2FS          # second extended file system (linux)
```

Das Linux-Filesystem *ext2*. Wenn man Linux-Partitionen verwenden möchten, etwa im Zuge eines Dual-Boot-Systems, sollte man ext2-Unterstützung einschalten. In der ext2-Unterstützung enthalten ist auch die Unterstützung von *ext3*.

```
file-system   MSDOSFS         # MS-DOS file system
file-system   NTFS            # Windows/NT file system
```

Erlauben Zugriff auf MS-DOS- (FAT) beziehungsweise NT-formattierte Platten und Disketten. Ab NetBSD 1.6 umfaßt die NTFS-Unterstützung auch das NTFS5-Datenträgerformat von Windows 2000.

```
file-system   CD9660          # ISO 9660 + Rock Ridge file system
```

Bindet das ISO-9660- und RockRidge-Dateisystem ein, das für CDs verwendet wird. Außer reinen ISO- und RockRidge-CDs werden auch Joliet-Extensions verarbeitet. Besitzt Ihr Rechner ein CD-ROM-Laufwerk, sollten Sie CD9660 unbedingt einbinden.

```
file-system   FDESC           # /dev/fd
```

Das File-Deskriptor-Dateisystem ermöglicht den Zugriff von Prozessen auf ihre Dateideskriptoren unter einer Verzeichnisstruktur. Dieses Dateisystem wird eigentlich nirgends produktiv verwendet; wenn Sie nicht planen, es einzusetzen, können Sie es bedenkenlos weglassen.

```
file-system   KERNFS          # /kern
```

Das Kernel-Dateisystem erlaubt es, diverse Kernelparameter als Dateien unter dem Mountpoint auszulesen. Alle diese Parameter sind auch über *sysctl(1)* erreichbar, und so ist das *kernfs* eher als Lehrstück als als wirklich sinnvoll zu betrachten. Kann ohne weiteres weggelassen werden.

```
file-system   NULLFS          # loopback file system
```

Durch das (verwirrenderweise) »Null« genannte Dateisystem können Teile eines Filesystems woanders gemountet werden. Das Ergebnis ist eine Mischung aus symbolischem Link und Kopie, denn die durch *nullfs* gemounteten Verzeichnisse erscheinen an ganz anderer Stelle im Dateisystem.

```
file-system   OVERLAY         # overlay file system
```

Ähnlich dem NullFS, dient eigentlich nur als Beispiel für ein Dateisystem.

```
file-system   PROCFS          # /proc
```

Das proc-Dateisystem ermöglicht den Zugriff auf diverse prozeßspezifische Parameter in Dateiform, ähnlich dem Kern-Dateisystem. Außer für Programme, die unter der Linux-Emulation laufen, wird es selten benötigt. Plant man allerdings, Linux-Binaries einzusetzen, sollte man sie »/proc«-gemountet haben.

```
file-system   UNION           # union file system
```

Experimentell. Das union-Dateisystem ist eine Abart des Null-Systems. Damit kann man Verzeichnisse »über« den Mountpoint legen, ähnlich einer Overhead-Folie über einer Zeichnung. Man sieht alles durch, kann jedoch durch Überschreiben Korrekturen und Veränderungen vornehmen. Diese Veränderungen werden im »oben« gemounteten Verzeichnis (der »Overheadfolie«) gespeichert, das darunterliegende Dateisystem wird nicht berührt (die »Zeichnung«). Damit kann man zum Beispiel ein »beschreibbares CD-ROM« realisieren.

```
file-system   CODA        # Coda File System; also needs vcoda (below)
```

Experimentell. CODA ist ein verteiltes Datesystem ähnlich wie NFS, jedoch mit speziellen Fähigkeiten bezüglich Caching. Es erlaubt auch den Zugriff, wenn die Netzwerkverbindung getrennt ist und aktualisiert später wieder selbständig die Dateien am Server. Benötigt auch das Pseudodevice *vcoda*.

```
#file-system PORTAL
```

Experimentell. Dieses Filesystem verbindet den Mountpoint mit dem Portal-Daemon, der es erlaubt, beliebige Programme beim Öffnen von Dateien in dieser Hierarchie auszuführen.

```
#file-system UMAPFS
```

Eine Variante des Null-FS, die zusätzlich noch die Umrechnung von User- und Gruppenberechtigungen erlaubt.

Während des Bootvorgangs muß der NetBSD-Kernel entscheiden, welche Konfiguration er an *init(8)* weitergibt. Durch die folgenden Eintragungen kann man zum Beispiel einen Kernel von einer CD starten, die Root-Partition jedoch dann von einer Festplatte oder über NFS mounten.

```
config        netbsd        root on ? type ?
#config       netbsd        root on sd0a type ffs
#config       netbsd        root on ? type nfs
```

Audio

Die folgenden Audio-Geräte werden von NetBSD unterstützt:

Device	Audiogerät
aria	Sierra Aria-basierend • Prometheus Aria 16 • Genoa AudioBahn 16 Pro • Diamond Sonic Sounds Rev A5 und Rev B2
auich	Intel ICH integrated AC'97 Audio
autri	Trident 4DWAVE-basierend AC'97 Audio • Trident 4DWAVE-DX/NX • SiS 7018 • ALi M5451
auvia	VIA VT82C686A integrated AC'97 Audio
clcs	Cirrus Logic CS4280
clct	Cirrus Logic CS4281
cmpci	C-Media CMI8338/8738
cms	Creative Music System
eap	Ensoniq AudioPCI • Ensoniq AudioPCI • Creative Labs SoundBlaster PCI
emuxki	Creative SBLive! and PCI512 • Creative Sound Blaster Live! • Sound Blaster PCI 512
esa	ESS Allegro-1 / Maestro-3 PCI Audio
esl	ESS 1688 AudioDrive
esm	ESS Maestro-1/2/2e PCI Audio Accelerator
eso	ESS Solo-1 PCI AudioDrive • TerraTec 128i PCI
ess	ESS Tech ES1887, ES1888, ES888
fms	Forte Media FM801

Tabelle 4.8 (Fortsetzung auf nächster Seite)

Device	Audiogerät
fms	Forte Media FM801
gus	Gravis Ultra Sound • Gravis UltraSound (GUS) • GUS MAX
guspnp	Gravis Ultra Sound PnP audio
neo	NeoMagic 256 AC'97 Audio • Meist in Laptops (Dell, HP)
pas	ProAudio Spectrum
pss	Personal Sound System • Cardinal Digital Sound Pro 16 • Orchid SoundWave 32
sb	SoundBlaster-kompatibles Audio • SoundBlaster • SoundBlaster Pro • SoundBlaster 16 • Jazz 16 • SoundBlaster AWE 32 und AWE 64
spkr	PC Lautsprecher Einfacher Ton-Generator für den standardmäßigen PC-Lautsprecher. Melodien (mit Pacman-Feeling :-) können durch Ausgabe von Textstrings nach »/dev/speaker« erzeugt werden.
sv	S3 SonicVibes-basierend • Turtle Beach Daytona
yds	Yamaha DS-1 PCI Audio
wss	Windows Sound System
ym	Yamaha OPL3-SA3 audio

Tabelle 4.8: Audiogeräte

An diesen Audio-Devices können – je nach Möglichkeiten der Hardware – verschiedene hardwareunabhängige Treiber angeschlossen werden:

```
audio*    at sb?
opl*      at sb?
midi*     at sb?          # SB1 MIDI port
```

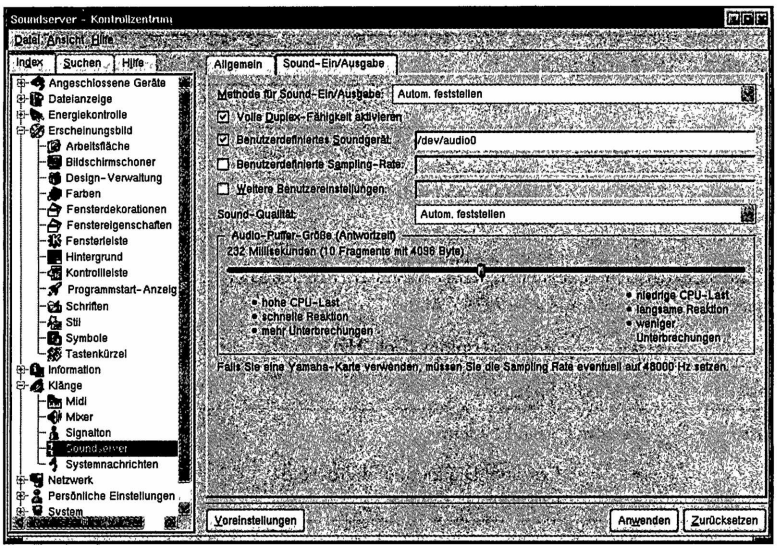


Abb. 4.2: Wenn man Audio unter KDE verwenden will, muß im Kontrollzentrum das entsprechende Device angegeben sein

Dabei ist *audio* der »normale« Audio-Treiber, über den zum Beispiel *.wav*-Dateien abgespielt werden. *opl* hingegen steuert Yamaha-OPL2- und OPL3-FM-Synthesizer an. Und schließlich bedient *midi* den MIDI-Port, der in manchen Soundkarten integriert ist.

USB

Der Universal Serial Bus wird schon vom GENERIC-Kernel für die meisten Anwendungen (Tastatur, Maus) unterstützt. Spezielle Geräte müssen jedoch explizit eingeschaltet werden.

Es gibt drei Arten von USB-Controllern: EHCI (nur experimenteller Treiber-Support), OHCI und UHCI. Davon wird üblicherweise nur ein Typ in einer Maschine vorkommen, die anderen können daher entfernt werden.

```
# The EHCI is not ready for prime time.
#ehci*    at pci?  dev ? function ?    # Enhanced Host Controller
ohci*     at pci?  dev ? function ?    # Open Host Controller
uhci*     at pci?  dev ? function ?    # Universal Host Controller (Intel)

# USB bus support
#usb*     at ehci?
usb*      at ohci?
usb*      at uhci?
```

Jeder an den USB angeschlossene Gerätetyp hat seinen eigenen Treiber, im folgenden eine Zusammenstellung der Geräte und der dazugehörigen Treiber.

Device	USB-Gerätetyp
uhidev	Human Interface Device, Obergruppe für sämtliche Eingabegeräte
ums	Mäuse
ukbd	Tastaturen
uhid	Sonstige (nicht näher definierte) Human Interface Devices[1]
ulpt	Drucker
uaudio	Audiogeräte (Lautsprecher, Digitizer)
urio	Diamond Multimedia Rio 500
uscanner	Scanner
usscanner	Scanner mit SCSI-Emulation, beispielsweise HP5300
uyap	Y@P Firmware Loader
udsbr	D-Link DSB-R100 USB Radio
umidi	USB MIDI
umass	SCSI oder ATAPI über USB (beispielsweise FlashCard-Leser) • FujiFilm FinePix1300 Digitalkamera • Imation SuperDisk, IOmega Zip 100 und IOmega Zip 250 • LaCie CD R/W und SanDisk ImageMate SDDR-31 • Siemens MP3-Player USB und Y-E Data FlashBuster Floppy • STMicroelectronics ST92163 Mass Storage Library Tester
uirda	IrDA Bridge • ACTiSYS ACT-IR2000U FIR-USB Adapter • Kawatsu KC-180 USB IrDA Device • Extended Systems XTNDAccess IrDA USB (ESI-9685)
ustir	SigmaTel STIr4200 USB/IrDA Bridge
ugen	USB Generic driver

Tabelle 4.9: Allgemeine USB-Geräte

[1] HID-Geräte sind ein besonders definierter Teil der USB-Definition. HID steht für »Human Device Interface« und charakterisiert speziell Eingabegeräte wie beispielsweise Tastatur und Maus. Ob ein Gerät zur Kategorie HID gehört, bestimmt aber der Hersteller.

Die seriellen Interfaces verbinden sich in der Form

```
ucom*   at   umct?
```

mit dem Treiber. Serielle Schnittstellen am USB werden über den *ucom*-Gerätetreiber verwaltet, der ein Interface wie zu einem normalen tty simuliert. Daher können normale Programme ohne Wissen über USB diese Schnittstellen genauso ansprechen, wie sie die seriellen Ports onboard verwenden würden.

Device	USB-Netzwerkkarten
aue	ADMtek AN986 Pegasus-basierte Adapter • Abocom DSB650TX • Billionton Systems USB100 • Corega FEther USB-TX • D-Link DSB-650, D-Link DSB-650TX, D-Link DSB-650TX-PNA • I/O DATA USB ETTX • Kingston KNU101TX • LinkSys USB100TX, LinkSys USB100H1, LinkSys USB10TA • Melco Inc. LU-ATX • SOHOware NUB100, SMC 2202USB
cue	Auf CATC USB-EL1201A basierende Adapter • Belkin F5U111, CATC Netmate und CATC Netmate II
kue	Kawasaki LSI KL5KUSB101B-basierte Adapter • 3Com 3c19250, 3Com 3c460 HomeConnect Ethernet USB • Abocom URE450 • ADS Technologies USB-10BT • Aox USB101 • ATen UC10T • Corega EtherUSB • D-Link DSB-650 • Entrega NET-USB-E45 • I/O Data USB-ET/T • Kawasaki USB101 • LinkSys USB10T • Netgear EA101 • Peracom USB Ethernet Adapter (3 models) • SMC 2102USB und SMC 2104USB
url	Auf Realtek RTL8150L basierende Adapters

Tabelle 4.10: USB-Ethernet-Adapter

Device	Serielle USB-Karte
uftdi	FTDI FT8U100AX Serieller Adapter • Inland UAS111 • QVS USC-1000
umct	MCT USB-RS232 Serieller Adapter • MCT USB-RS232 Converter 25-Pin Modell U232-P25 • MCT USB-RS232 Converter 9-Pin Modell U232-P9 • Sitecom USB-RS232 Converter • D-Link DU-H3SP USB BAY
uplcom	I/O DATA USB-RSAQ2 Serieller Adapter • IOGEAR UC-232A • I/O DATA USB-RSAQ und I/O DATA USB-RSAQ2 • PLANEX USB-RS232 URS-03 • BAFO BF-800 und BAFO BF-810 • ELECOM UC-SGT
uvscom	SUNTAC Slipper U VS-10U Serieller Adapter
umodem	Modems
uvisor	USB Handspring Visor

Tabelle 4.11: Serielle Schnittstellen am USB

Pseudo-Devices

```
pseudo-device   ccd   4          # concatenated/striped disk devices
```

Der CCD-Treiber (Concatenated Disk) erlaubt die Verbindung mehrerer Platten zu einer großen, virtuellen Disk. Dabei werden die Daten im Striping-Verfahren über die Platten verteilt, es entspricht also einem RAID-0[1]. Falls Sie nur RAID-0-Funktionalität benötigen, ist CCD gegenüber RAIDframe (siehe unten) die bessere Wahl, da einfacher und kompakter.

Die Zahl 4 gibt die Maximalanzahl der verfügbaren CCD-Pseudoplatten an.

[1] RAID-0 oder »Striping« ist die Zusammenfassung mehrerer Disks, um eine größere Platte zu erhalten. Dabei werden die Daten in kleinen Teilen (den Stripes) auf die verschiedenen Disks verteilt, um die Zugriffszeiten zu minimieren (während eine Platte Daten liest, können die anderen die Schreib-/Leseköpfe positionieren). Doch Vorsicht, RAID-0 bietet keinerlei Datensicherheit: Ist eine Platte defekt, sind die Daten in ihrer Gesamtheit nicht mehr lesbar!

```
pseudo-device  raid            8 # RAIDframe disk driver
options        RAID_AUTOCONFIG    # auto-configuration of RAID components

#options       RF_INCLUDE_EVENODD=1
#options       RF_INCLUDE_RAID5_RS=1
#options       RF_INCLUDE_PARITYLOGGING=1
#options       RF_INCLUDE_CHAINDECLUSTER=1
#options       RF_INCLUDE_INTERDECLUSTER=1
#options       RF_INCLUDE_PARITY_DECLUSTERING=1
#options       RF_INCLUDE_PARITY_DECLUSTERING_DS=1
```

RAIDframe ist die generische RAID-Lösung von NetBSD, die die RAID-Levels 0, 1, 4, 5 und einige weitere Varianten zur Verfügung stellt. Mit RAIDframe kann man Platten spiegeln und zusammenfassen, das ganze auch noch geschachtelt (zum Beispiel ein RAID-1 von RAID-5).

Die Zahl 8 auf der Pseudo-Device-Zeile gibt die Anzahl der maximal verfügbaren RAIDframe-Pseudoplatten an.

RAID_AUTOCONFIG schaltet die automatische Erkennung und Konfiguration von RAID-Komponenten ein, damit kann zum Beispiel auch das root-Filesystem auf eine durch RAID vor Plattenausfall geschützte Partition gelegt werden.

Die oben auskommentierten Optionen schalten manche der üblicherweise nicht verwendeten Features, die standardmäßig nicht eingebunden werden, explizit ein. Sie sind aber für den Normalanwender, der sich nicht intensiv mit dem Innenleben von RAIDframe beschäftigen will, nicht relevant.

```
pseudo-device  md  1     # memory disk device (ramdisk)
```

md ist eine RAM-Disk, die eine virtuelle Platte zur Verfügung stellt, die vollständig im Speicher gehalten wird. Wird von den Installationsdisketten verwendet, um eine Root-Platte inklusive Inhalt zu erzeugen.

```
pseudo-device   vnd  4   # disk-like interface to files
```

vnd erlaubt den Zugriff auf Dateien so, als wären sie Platten. Das braucht man, wenn beispielsweise ein ISO-Image einer CD direkt gemountet werden soll[1], ohne es zuerst auf CD zu brennen. Der angegebene Parameter ist die maximale Anzahl von gleichzeitig verwendeten *vnd*-Geräten. Siehe auch *vnconfig(8)*.

[1] Entspricht dem *loop*-Device unter Linux.

Virtuelle Mounts von Images

Wenn man ein ISO-Image einer CD vorliegen hat und darauf zugreifen will, muß man es nicht zwangsläufig auf einen Rohling brennen. Man nutzt man das Loopback-Device als Pseudo-Mountpunkt für das Image. Unter NetBSD erfolgt das Einbinden mit

```
vnconfig /dev/vnd0d /<Pfad>/<ISO-Image-Name>
mount -t cd9660 /dev/vnd0d /<Mount-Verzeichnis>
```

Alternativ kann man anstelle von *mount -t cd9660* den Befehl *mount_cd9660* eingeben.

Es empfiehlt sich, das Image von vornherein schreibgeschützt einzubinden. Das wird bei *mount_cd9660* mit *-o rdonly* vorgenommen. Beim generischen *mount* wird hierfür der Parameter *-r* genommen. Bei *mount_cd9660* hat *-r* eine andere Bedeutung, nämlich daß RockRidge-Extensions nicht verwendet werden. Dann fehlen allerdings die langen Dateinamen. Davon sollte man natürlich Abstand nehmen.

Selbstverständlich funktioniert die Aktion mit beliebigen ISO-Images im ISO-, Rockridge- und Joliet-Format. Wenn man die Aktionen nicht als *root*, sondern als normaler Anwender durchführen will, müssen aber vorher die Dateirechte stimmen oder der Mount-Punkt unter das Heimatverzeichnis gelegt werden.

Den Image-Mount löst man genauso wieder in zwei Schritten auf.

```
umount /<pfad>
vnconfig -u /dev/vnd0d
```

Natürlich ist das Konfigurieren und Einbinden des Loopback-Devices nicht auf CD-ROM-Images beschränkt. Das Einbinden anderer Datenträger-Images wie beispielsweise von Disketten ist genauso möglich. Nähere Informationen dazu stehen auf der Manpage von *vnconfig*. NetBSD bemängelt außerdem beim Mount die Tatsache, daß CD-ISO-Images kein Disklabel besitzen. Diese Meldung kann aber getrost ignoriert werden.

```
pseudo-device   pty      # pseudo-terminals
```

Bindet den Pseudoterminal-Treiber ein. Pseudoterminals werden zum Beispiel für Shells verwendet, ohne diesen Treiber ist das Einloggen unmöglich. Unbedingt eingeschaltet lassen!

```
pseudo-device   tb   1    # tablet line discipline
```

Treiber für Grafik-Tabletts. Unterstützt werden dabei die Hersteller Hitachi, GTCO, und Polhemus.

```
pseudo-device   sequencer  1    # MIDI sequencer
```

Treiber für MIDI-Sequencer

```
# rnd works; RND_COM does not on port i386 yet.
pseudo-device   rnd             # /dev/random and in-kernel generator
#options        RND_COM         # use "com" randomness as well (BROKEN)
```

Der *rnd*-Treiber wird verwendet, um qualitativ hochwertige Zufallszahlen zu generieren. Dazu sammelt *rnd* Statistikdaten von diversen anderen Devices, um daraus Zufallsfolgen zu erzeugen. Diese Zufallsdaten können via */dev/random* und */dev/urandom* ausgelesen werden. Wie der Kommentar andeutet, ist *RND_COM* (Sammeln von Statistiken von der seriellen Schnittstelle) derzeit nicht funktionsfähig.

```
# a pseudo device needed for Coda       # also needs CODA (above)
pseudo-device       vcoda       4       # coda minicache <-> venus comm.
```

Sonstige Geräte

Im folgenden sind die Geräte aufgelistet, die sich nicht einem der vorhergehenden Abschnitte zuordnen lassen.

Floppies mounten

Das Einbinden von Disketten erfolgt unter NetBSD genauso wie jeder andere Mount. Es ist aber zweckmäßig, zwei getrennte Mountpunkte für Disketten, die im FFS-Format formatiert sind, und solche, die dem Datenaustausch dienen und das MS-DOS-Dateisystem besitzen, anzulegen. Am bequemsten ist es, getrennte Verzeichnisse, beispielsweise */mnt/A* für FAT-Disketten und */mnt/floppy* für FFS-Disketten zu verwenden. Dafür trägt man in die Datei */etc/fstab* die beiden Zeilen ein:

```
/dev/fd0a       /mnt/floppy     ffs     rw,noauto       0 0
/dev/fd0a       /mnt/A          msdos   rw,noauto       0 0
```

Nun wird je nach Format des Datenträgers nur entweder *mount /mnt/A* oder *mount /mnt/floppy* angegeben. Das System holt sich dann das passende Dateisystem aus der Beschreibungdatei.

Der standardmäßige Floppy-Controller wird durch *fdc* bedient, die Laufwerke dann als *fd0* und *fd1*.

```
# ISA floppy
fdc0    at isa? port 0x3f0 irq 6 drq 2   # standard PC floppy controllers
fd*     at fdc? drive ?                  # the drives themselves
```

Auf modernen Motherboards sind oft auch Geräte zur Überwachung der Hardware (Drehzahl der Lüfter, Temperatur der CPU) integriert. Die verbreitetsten Vertreter dieser Familie sind:

```
# Hardware monitors
# LM7[89] and compatible hardware monitors
#lm0      at isa?    port 0x290          # other common ports: 0x280, 0x310

# VIA VT82C686A hardware monitor
#viapm*   at pci? dev ? function ?
#viaenv* at viapm?
```

Mäuse – besonders ältere Modelle – werden manchmal von speziellen Schnittstellenkarten angesteuert, die dann auch eigene Treiber benötigen. Moderne Mäuse, die über den PS2-Port oder USB angeschlossen sind, benötigen die folgenden Treiber nicht.

```
# ISA busmice
lms0        at isa? port 0x23c irq 5       # Logitech bus mouse
lms1        at isa? port 0x238 irq 5
mms0        at isa? port 0x23c irq 5       # Microsoft InPort mouse
mms1        at isa? port 0x238 irq 5
wsmouse*    at lms? mux 0
wsmouse*    at mms? mux 0
```

wscons und die Maus

Der Anschluß der Maus hat unter NetBSD keine Auswirkung auf die Definition des Devices, da die Gerätedatei »/dev/wsmouse« auf das eigentliche Gerät gemappt wird. Es muß nur darauf geachtet werden, daß die entsprechende Schnittstelle im Kernel definiert ist. Dann wird beispielsweise X so konfiguriert, daß als Maus-Schnittstelle »/dev/wsmouse« und als Mausprotokoll »wsmouse« definiert werden. Selbst beim Wechsel der Maus – beispielsweise von PS/2 auf USB – muß anschließend die Mausschnittstelle nicht neu konfiguriert werden.

Plattformspezifische Optionen (Mac68k)

von Jörg Braun

Zugegeben ist der Macintosh und schon gerade der auf Basis Motorola 680x0 nicht eine der bedeutendsten Plattformen. Die Kernel-Konfigurationsdatei ist aber sehr überschaubar. Sie soll deshalb hier als Beispiel für Nicht-Intel-Systeme dienen und zeigen, daß zwar die Devices anders heißen, prinzipiell aber die Konfiguration bei allen Ports ähnlich vorgenommen wird.

Vorausgeschickt werden muß, daß es eine Tortur ist, auf einem Rechner mit einer Taktfrequenz von 25 bis 33 MHz den Kernel zu übersetzen. Es empfiehlt sich deshalb, die Kernel-Konfiguration auf einer schnelleren Maschine cross zu übersetzen und dann die neue Kerneldatei auf den langsamen Rechner zu übertragen.

Ein Problem bei NetBSD 1.6 ist, daß es keine Möglichkeit gibt, elegant den Kernel direkt für eine andere Plattform zu übersetzen. Am einfachsten erfolgt das mit einem Komplettlauf der Distributionstools (Kapitel 16). Kurz nach Erscheinen von NetBSD 1.6 wurde aber am 21.10.2002 ein neuer Schalter der Datei »/usr/src/build.sh« in den CURRENT-Tree eingeführt, der das Crosskompilieren des Kernels ermöglicht. Er steht in der Release ab NetBSD 1.6.1 zur Verfügung. Mit *-k kernel* wird der angegebene Kernel konfiguriert und gebaut.

Alte Maschinen sollten mit Usern nicht überlastet werden, weshalb bei mac68k *maxusers* deshalb auf 16 begrenzt ist:

```
maxusers        16              # estimated number of users
```

Die CPU-Unterstützung ist ähnlich wie bei der Intel-Plattform. Auch hier besteht die Auswahl unter mehreren unterschiedlichen Generationen und auch hier muß mindestens eine CPU definiert sein:

```
# CPU support.  At least one is REQUIRED.
options         M68040
options         M68030
options         M68020          # Note:  must have 68851 PMMU
```

Probleme bestehen auf Systemen ohne Fließkommaeinheit. Sie wird von NetBSD zwingend vorausgesetzt.

```
# CPU-related options.
options         FPSP
options         FPU_EMULATE
```

Hier sind nicht alle Optionen aufgelistet. Die plattformunabhängigen sind bereits beschrieben. Hier sind nur die macintosh-typischen dargestellt:

```
# Mac-specific options
#options  ALTXBUTTONS     # Map Opt-{1,2,3} to mouse buttons
#options  DISABLE_EXT_CACHE # If really paranoid, disable IIci ext. cache
options   GRF_COMPAT      # Include grf compatibility code
#options  MRG_ADB         # Use ROM-based ADB driver
options   ZS_CONSOLE_ABORT

options         NFS_BOOT_DHCP

# Kernel root file system and dump configuration.
config          netbsd  root on ? type ?
```

Die Hardware selbst ist bei jedem Port mit eigenen Geräten abgebildet:

```
#
# Device configuration
#

mainbus0 at root

fpu0 at mainbus?      # Floating-Point Coprocessor support


# Basic Bus Support

# On-board I/O bus support
obio0 at mainbus?

# NuBus support
nubus0 at mainbus?


# Console Devices

# Apple Desktop Bus interface
adb0    at obio?
aed*    at adb?                         # ADB event device
akbd*   at adb?                         # ADB keyboard
ams*    at adb?                         # ADB mouse

# Basic frame buffer support
intvid0 at obio?                        # Internal video hardware
macvid* at nubus?                       # NuBus video card

# Device-independent frame buffer interface
macfb*  at intvid?
macfb*  at macvid?
```

```
# Workstation Console devices
wsdisplay* at macfb? console ?
wskbd*  at akbd? console ?
wsmouse* at ams?


# Serial Devices

# On-board serial interface
zsc0    at obio?
zstty*  at zsc? channel ?

# SCSI Controllers and Devices

# SCSI controllers
# XXX - use only one of ncrscsi or sbc
ncrscsi0 at obio?                       # SCSI NCR 5380
#sbc0    at obio? flags 0x1             # MI SCSI NCR 5380
esp0    at obio?                        # SCSI NCR 53C9x
esp1    at obio?                        # SCSI NCR 53C9x

# .... gekürzt

# IDE controller and devices
wdc*    at obio? flags 0x1000
wd*     at wdc? channel ? drive ?

# Miscellaneous mass storage devices

# IWM floppy disk controller
iwm0    at obio?                        # Sony driver (800K GCR)
fd*     at iwm? drive ?

# Network Interfaces

# On-board Ethernet controllers
sn*     at obio?                        # SONIC-based (DP83932, DP83916)
mc*     at obio?                        # MACE-based

# NuBus Ethernet controllers
ae*     at nubus?                       # DP8390-based
sn*     at nubus?                       # SONIC-based (DP83932, DP83916)
sm*     at nubus?                       # SMC 91cxx-based

nsphy*  at mii? phy ?
ukphy*  at mii? phy ?
```

```
# Audio Devices

# On-board audio hardware
asc0    at obio?                        # ASC/EASC audio
```

Auf dieser Plattform ist die Hardware sehr viel limitierter. Man findet sich, wenn man die beiden Plattformen i386 und mac68k vergleicht, sehr schnell zurecht. Allein das ist ein Grund, wenn man verschiedene Hardware verwenden will, auf das selbe Betriebssystem zurückzugreifen.

4.3 Die Kernel-Konfiguration für ISDN

von Karl Uwe Lockhoff

Seit der Version 1.6 unterstützt NetBSD ISDN nicht mehr über das problematische externe Einbinden des I4B-Systems[1]. ISDN ist aber nicht im generischen NetBSD-Kernel enthalten. Wenn Sie eine ISDN-Karte verwenden wollen, müssen Sie also einen neuen Kernel bauen. Verwenden Sie beim i386-Port als Grundlage die Kernel-Konfigurationsdatei »GENERIC_ISDN«. Es handelt sich dabei um einen nur um die ISDN-Geräte erweiterten generischen Kernel.

4.3.1 Die Grundkonfiguration

Die Grundkonfiguration für ISDN ist einfach durchzuführen. Hierzu müssen nur die folgenden Zeile in die Kernelkonfiguration aufgenommen werden:

```
pseudo-device   isdn              # Kommunikation zwischen
                                  # Kernel und isdnd
pseudo-device   isdntrc         2 # Geräte zum Debuggen der Verbindung
pseudo-device   isdnctl           # Gerät zur Steuerung der Verbindung
pseudo-device   isdnbchan       4 # Zugriff auf den B-Kanal
pseudo-device   isdntel         2 # Telefongerät
options         IPR_VJ            # Van-Jacobsen-Kompression
pseudo-device   irip            2 # Netzwerktreiber für
                                  # IP über HDLC ISDNnetwork
pseudo-device   ippp            2 # synchrones PPP
```

Das Pseudo-Device *isdn* übernimmt hierbei die Hauptaufgabe, die Kommunikation mit dem Daemon *isdnd,* und ist zwingend erforderlich. Das Gerät *isdntrc* ist optional. Es dient dazu, sich den Datenstrom auf den ISDN-Kanälen mit dem Kommando *isdntrace* anzuzeigen zulassen.

Mit *isdnctl* wird die Verbindung gesteuert. Es ist technisch gesehen die Schnittstelle zum

[1] »I4B« steht für »ISDN for BSD«. Es wurde ursprünglich als Zusatzpaket für die verschiedenen BSD-Systeme angeboten und ursprünglich für FreeBSD entwickelt.

D-Kanal. Es werden mit Hilfe dieses Geräts also solche Kommandos wie das Auflegen durchgeführt. Damit auch Daten über ISDN versendet werden können, ist noch der Zugriff auf den B-Kanal erforderlich. Er wird durch *isdnbchan* zur Verfügung gestellt. Hierbei sind pro B-Kanal zwei Geräte zu konfigurieren, jeweils einer für jede Richtung.

Mit diesen Erstellungen könnte man schon eine Kommunikation bewerkstelligen. Allerdings ist dies programmtechnisch schwierig zu realisieren. Deshalb gibt es noch drei verschiedene Geräte, mit deren Hilfe die drei wichtigsten Protokolle zum Datentransfer durchgeführt werden: für die Sprachtelephonie *isdntel*, für IP über HDLC *irip* und für synchrones PPP *ippp*. Es werden pro B-Kanal immer zwei Geräte benötigt.

Die Option *IPR_VJ* stellt eine Kompression auf dem Netzwerk bereit.

4.3.2 ISDN-Karten

Jetzt ist noch die Verbindung zur eigentlichen Hardware, der ISDN-Karte, herzustellen:

```
options ISICISA_TEL_S0_8
isic0 at isa? iomem 0xd0000 irq 5
```

Diese Angabe soll hier nur als ein Beispiel dienen. Für die genaue Konfiguration einer bestimmten Karte sind weitere Kenntnisse notwendig. In der folgenden Übersicht sind einzelne Karten, deren Eigenschaften und jeweils ein Beispiel für einen Eintrag in der Kernelkonfiguration zusammengefaßt.

ISA-Karten

Teles S0/8, Dr. Neuhaus Niccy 1008, Creatix ISDN-S0/8

Die Angabe eines Ports ist bei dieser Karte nicht notwendig. Als Interrupt ist der Bereich von 2 bis 7 gültig. Als Speicherbereich wird ein 4-Kbyte-Speicher im Bereich von 0xA0000 bis 0xDF000 verwendet.

Beispiel:

```
options ISICISA_TEL_S0_8
isic0 at isa? iomem 0xd0000 irq 5
```

Teles S0/16, Creatix ISDN-S0, Dr. Neuhaus Niccy 1016

Diese Karten haben einen Jumper, mit dem die I/O-Basisadresse festgelegt wird: 0xd80, 0xe80 oder 0xf80. Alle weiteren Angaben werden bei der Initialisierung in diesen Port geschrieben. Der Interrupt ist 2, 3, 4, 5, 10, 11, 12 oder 15. Die möglichen Speicherbereiche sind 0xC0000, 0xC2000, 0xC4000, 0xC6000, 0xC8000, 0xCA000, 0xCC000, 0xCE000, 0xD0000, 0xD2000, 0xD4000, 0xD6000, 0xD8000, 0xDA000, 0xDC000 und 0xDE000.

Anmerkung: Die Jumper sind zwar mit den Portadressen 0xd80, 0xe80 und 0xf80 beschriftet, aber es wird ein immer ein weiterer Bereich belegt: 0x180, 0x280 oder 0x380.

Beispiel:

```
options ISICISA_TEL_S0_16
isic0 at isa? port 0xd80 iomem 0xd0000 irq 5
```

Teles S0/16.3

Diese Karten kommt ohne *iomem*-Zeile aus. Der Interrupt ist 2, 5, 9, 10, 12 oder 15.

Anmerkung: Die Jumper sind zwar mit den Portadressen 0x180, 0xe80 und 0x380 beschriftet, aber es wird ein immer ein weiterer Bereich belegt: 0xd80, 0xe80 oder 0xf80.

Beispiel:

```
options ISICISA_TEL_S0_16_3
isic0 at isa? port 0xd80 irq 5
```

AVM A1, AVM Fritz!Card

Diese Karten besitzen einen Jumper, um die I/O-Basisadresse anzugeben: 0x200, 0x240, 0x300 oder 0x340. Der Interrupt ist 3, 4, 5, 6, 7, 10, 11, 12 oder 15. Diese Karten brauchen keinen *iomem*-Eintrag.

Beispiel:

```
options ISICISA_AVM_A1
isic0 at isa? port 0x340 irq 5
```

Teles S0/16.3 PnP

Als I/O-Basisadresse sind 0x580, 0x500 oder 0x680 möglich. Der Interrupt ist 3, 5, 7, 10, 11 oder 12. Die Karte wird automatisch vom Kernel konfiguriert.

Beispiel:

```
options ISICPNP_TEL_S0_16_3_P
isic* at isapnp?
```

Creatix ISDN-S0 P&P

Als I/O-Basisadresse sind 0x180, 0x180 oder 0x100 möglich. Der Interrupt ist 3, 5, 7, 10, 11 oder 12. Die Karte wird automatisch vom Kernel konfiguriert.

Beispiel:

```
options ISICPNP_CRTX_SO_P
isic* at isapnp?
```

3Com USRobotics Sportster ISDN TA intern, Stollmann Tina pp

Als I/O-Basisadresse sind 0x200, 0x208, 0x210, 0x218, 0x220, 0x228, 0x230, 0x238, 0x240, 0x248, 0x250, 0x258, 0x260, 0x268, 0x270 oder 0x278 möglich. Der Interrupt ist 3, 5, 7, 10, 11 oder 12.

Anmerkung: Dieser Karte besitzt eine ungewöhnliche Adreßdekodierung. Trotzdem ist die Unterstützung dieser Karte gut, da der Hersteller die technischen Unterlagen herausgegeben hat.

Beispiel:

```
options ISICISA_USR_STI
isic0 at isa? port 0x268 irq 5
```

Dr. Neuhaus Niccy Go@

Als I/O-Basisadresse ist der Bereich von 0x200 bis 0x3e0 möglich. Der Interrupt ist 3, 4, 5, 9, 10, 11, 12 oder 15. Diese Karte wird automatisch vom Kernel konfiguriert.

Beispiel:

```
options ISICPNP_DRN_NGO
isic* at isapnp?
```

Sedlbauer Win Speed

Als I/O-Basisadresse ist der Bereich von 0x100 bis 0x3f0 möglich. Der Interrupt ist 3, 4, 5, 7, 10, 11, 12, 13 oder 15. Diese Karte wird automatisch vom Kernel konfiguriert.

Beispiel:

```
options ISICPNP_SEDLBAUER
isic* at isapnp?
```

ELSA QuickStep 1000pro (ISA)

Als I/O-Basisadresse ist der Bereich von 0x160 bis 0x360 möglich. Der Interrupt ist 3, 4, 5, 7, 10, 11, 12 oder 15. Diese Karte wird automatisch vom Kernel konfiguriert.

Beispiel:

```
options ISICPNP_ELSA_QS1ISA
isic* at isapnp?
```

ITK ix1 micro

Von dieser Karte liegen keine Informationen vor. Beispiel:

```
options ISICISA_ITKIX1
isic0 at isa? port 0x398 irq 10
```

DynaLink IS64PH

Von dieser Karten liegen keine Informationen vor. Beispiel:

```
options ISICPNP_DYNALINK
isic* at isapnp?
```

PCI-Karten

ELSA QuickStep 1000pro-PCI

Diese Karte wird automatisch vom Kernel konfiguriert. Beispiel:

```
isic*  at pci?
```

Winbond W6692

Diese Karte ist autokonfigurierend, besitzt aber einen eigenen Treiber. Beispiel:

```
iwic*  at pci?
```

AVM Fritz!PCI card

Diese Karte ist autokonfigurierend, besitzt aber einen eigenen Treiber. Beispiel:

```
ifpci*  at pci?
```

AVM Fritz!PCI V2 card

Diese Karte ist autokonfigurierend, besitzt aber einen eigenen Treiber. Beispiel:

```
ifritz* at pci?
```

PCMCIA-Karten

Alle PCMCIA-Karten sind autokonfigurierend. Somit muß der richtige Treiber im Kernel sichergestellt sein. Im folgenden sind nur die für die Karten notwendigen Eintragungen in die Konfigurationsdatei angegeben:

AVM Fritz!Card PCMCIA

```
options ISICPCMCIA_AVM_A1
isic* at pcmcia? function ?
```

ELSA MicroLink ISDN/MC

```
options ISICPCMCIA_ELSA_ISDNMC
isic* at pcmcia? function ?
```

ELSA MicroLink MC/all

```
options ICISPCMCIA_ELSA_MCALL
isic* at pcmcia? function ?
```

Sedlbauer speed star II

```
options ICISPCMCIA_SBSPEEDSTAR2
isic* at pcmcia? function ?
```

ISDN-Karten für den Amiga

BSC ISDN Master (2092/64), BSC ISDN MasterII (2092/65), ITH ISDN MasterII (5000/1), VMC ISDN Blaster (5001/1), Zeus ISDN Link (2189/3)

Diese Karten werden vom Zorro Bus Kernel System konfiguriert. Die ISDN-Funktionen dieser Karten liegen danach an bekannten Adressen.

Anmerkungen: Der Interrupt der Karte sollte als IPL2 und nicht wie unter AmigaOS üblich als IPL6 eingestellt werden. Die ITH ISDN MasterII arbeitet im DraCo Zorro Bus nicht. Dies ist ein generelles Problem und hat nichts mit NetBSD zu tun. Beispiel:

```
aster* at zbus?
isic* at aster? port ?
```

Individual Computers ISDN Surfer (4626/5 serno 0)

Diese Karten wird vom Zorro Bus Kernel System konfiguriert.

Anmerkung: Der NetBSD-Treiber arbeitet mit IPL2. Der AmigaOS-Treiber weist der Karte den Interrupt IPL6 zu. Hierdurch kann das System kurz nach dem Booten stehen bleiben. In diesem Fall ist NetBSD mit dem Bootblock oder über *loadbsd* ohne vorher den AmigaOS-Treiber zu aktivieren zu booten. Beispiel:

```
aster* at zbus?
isic* at aster? port ?
```

Diese Liste ist unbedingt als vollständig anzusehen. Es ist denkbar, daß einige der oben genannten Karten auch unter anderen Bezeichnungen, als Nachbauten oder OEM-Produkte auf dem Markt vertrieben werden.

Es gibt auch sogenannte ISDN-Terminaladapter, kurz »TA«. Sie verhalten sich wie Modems und werden auch als solche gehandhabt.

4.4 Das Programm sysctl

von Jörg Braun

Nicht jede Systemänderung verlangt zwangsläufig ein Neukompilieren des Kernels. Zahlreiche Einstellungen lassen sich auch mit dem Programm *sysctl* ändern. Die entsprechenden Kernelvariablen listet Tabelle 4.13 auf. Mögliche Parameter beim Aufruf des Programms sind:

Parameter	Beschreibung
-n	Den Namen bei der Ausgabe unterdrücken (für Shellskripte). Es handelt sich um einen Zusatzparameter für die anderen Aufrufe.
name ...	Anzeigen der Werte für die benannte Kernelvariable.
-w name=wert	Zuweisen (»write«) des Werts *wert* für die Kernelvariable *name*.
-a	Auflisten aller Werte (Strings und Zahlen).
-A	Anzeigen aller Geräte des Management Info Bus (MIB) inklusive Tabellen.
[-q]	Nur im Fehlerfall Ausgaben tätigen (»quiet«).
-f file	Angabe einer Kerneldatei für die Arbeiten.

Tabelle 4.12: Die Parameter von sysctl. -q und -n sind nur in Kombination mit anderen Parametern zulässig

Die Änderung von Parametern zur Laufzeit hat den Vorteil, daß man nicht gleich einen neuen Kernel bauen muß. Man trägt dazu in die Konfigurationsdatei »/etc/sysctrl.conf« nur den neuen Wert ein. Einige Variablen wie die für den Uhrzeit-Offset lassen sich mit

dem GNU Debugger (gdb) für den aktuellen Kernel permanent ändern. Dazu darf der Kernel aber nicht komprimiert sein[1]. Man ruft dazu den *gdb* mit dem Parameter *--write /netbsd* auf und gibt den Namen der Variable und den neuen Wert an, beispielsweise

```
kern.rtc_offset=-60
```

Beim Beenden des Debuggers mit *quit* wird die Kerneldatei zurückgeschrieben.

Name der Kernel-Variable	Typ	änderbar?
Debugger		
ddb.fromconsole	Integer	ja
ddb.lines	Integer	ja
ddb.maxoff	Integer	ja
ddb.maxwidth	Integer	ja
ddb.onpanic	Integer	ja
ddb.radix	Integer	ja
ddb.tabstops	Integer	ja
Hardware		
hw.alignbytes	Integer	nein
hw.byteorder	Integer	nein
hw.disknames	String	nein
hw.diskstats	Struktur	nein
hw.machine	String	nein
hw.machine_arch	String	nein
hw.model	String	nein
hw.ncpu	Integer	nein

Tabelle 4.13 (Fortsetzung auf nächster Seite)

[1] Bei Platzproblemen im Root-Dateisystem ist es recht praktisch, wenn man den Kernel mit *gzip* packt. Dazu ruft man *gzip -9* mit dem Namen des Kernels auf. Die daraus entstandene Datei »/netbsd.gz« benennt man anschließend wieder in »/netbsd« um. Nur Patch-Arbeiten am Kernel funktionieren so nicht mehr! Wenn man Variablen im Kernel ändern will, muß man ihn vorher mit *gunzip* entpacken und anschließend neu komprimieren.

Name der Kernel-Variable	Typ	änderbar?
hw.pagesize	Integer	nein
hw.physmem	Integer	nein
hw.usermem	Integer	nein
hw.cnmagic	String	ja
Kernel		
kern.argmax	Integer	nein
kern.autonicetime	Integer	ja
kern.autoniceval	Integer	ja
kern.boottime	Struktur	nein
kern.chown_restricted	Integer	nein
kern.clockrate	Struktur	nein
kern.defcorename	String	ja
kern.fsync	Integer	nein
kern.hostid	Integer	ja
kern.hostname	String	ja
kern.iov_max	Integer	nein
kern.job_control	Integer	nein
kern.link_max	Integer	nein
kern.login_name_max	Integer	nein
kern.logsigexit	Integer	ja
kern.max_canon	Integer	nein
kern.max_input	Integer	nein
kern.maxfiles	Integer	ja
kern.maxpartitions	Integer	nein
kern.maxproc	Integer	ja
kern.maxptys	Integer	ja, speziell
kern.maxvnodes	Integer	nur erhöhen

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernel-Variable	Typ	Änderbar?
kern.mapped_files	Integer	nein
kern.memlock	Integer	nein
kern.memlock_rage	Integer	nein
kern.memory_protection	Integer	nein
kern.mbuf.mblowat	Integer	ja
kern.mbuf.mcllowat	Integer	ja
kern.mbuf.mclsize	Integer	nein
kern.mbuf.msize	Integer	nein
kern.mbuf.nmbclusters	Integer	nur vergrößern
kern.msgbufsize	Integer	nein
kern.name_max	Integer	nein
kern.ngroups	Integer	nein
kern.nein_trunc	Integer	nein
kern.osrelease	String	nein
kern.osrevision	Integer	nein
kern.ostype	String	nein
kern.path_max	Integer	nein
kern.pipe.maxkvasz	Integer	ja
kern.pipe.maxloankvasz	Integer	ja
kern.pipe.maxbigpipes	Integer	ja
kern.pipe.nbigpipes	Integer	nein
kern.pipe.kvasize	Integer	nein
kern.posix1version	Integer	nein
kern.rawpartition	Integer	nein
kern.rtc_offset	Integer	Im Debugger
kern.saved_ids	Integer	nein
kern.securelevel	Integer	nur erhöhen
kern.synchronized_io	Integer	nein

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernel-Variable	Typ	Ist änderbar?
kern.sysvmsg	Integer	nein
kern.sysvsem	Integer	nein
kern.sysvshm	Integer	nein
kern.tkstat.nin	Quad	nein
kern.tkstat.nout	Quad	no
kern.tkstat.cancc	Quad	no
kern.tkstat.rawcc	Quad	no
kern.vdisable	Integer	nein
kern.version	String	nein
Konsole		
machdep.console_device	dev_t	nein
Netzwerk		
net.inet.icmp.maskrepl	Integer	ja
net.inet.icmp.errppslimit	Integer	ja
net.inet.icmp.rediraccept	Integer	ja
net.inet.icmp.redirtimeout	Integer	ja
net.inet.ip.allowsrcrt	Integer	ja
net.inet.ip.anonportmax	Integer	ja
net.inet.ip.anonportmin	Integer	ja
net.inet.ip.directed-broadcast	Integer	ja
net.inet.ip.forwarding	Integer	ja
net.inet.ip.forwsrcrt	Integer	ja
net.inet.ip.maxfragpacket	Integer	ja
net.inet.ip.lowportmax	Integer	ja
net.inet.ip.lowportmin	Integer	ja
net.inet.ip.mtudisc	Integer	ja
net.inet.ip.mtudisctimeout	Integer	ja

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernelvariable	Typ	änderbar?
net.inet.ip.redirect	Integer	ja
net.inet.ip.subnetsarelocal	Integer	ja
net.inet.ip.ttl	Integer	ja
net.inet.ipsec.ah_cleartos	Integer	ja
net.inet.ipsec.ah_net_deflev	Integer	ja
net.inet.ipsec.ah_offsetmask	Integer	ja
net.inet.ipsec.ah_trans_deflev	Integer	ja
net.inet.ipsec.def_policy	Integer	ja
net.inet.ipsec.dfbit	Integer	ja
net.inet.ipsec.ecn	Integer	ja
net.inet.ipsec.esp_net_deflev	Integer	ja
net.inet.ipsec.esp_trans_deflev	Integer	ja
net.inet.ipsec.inbound_call_ike	Integer	ja
net.inet.tcp.ack_on_push	Integer	ja
net.inet.tcp.compat_42	Integer	ja
net.inet.tcp.cwm	Integer	ja
net.inet.tcp.cwm_burstsize	Integer	ja
net.inet.tcp.init_win	Integer	ja
net.inet.tcp.keepcnt	Integer	ja
net.inet.tcp.keepidle	Integer	ja
net.inet.tcp.keepintvl	Integer	ja
net.inet.tcp.log_refused	Integer	ja
net.inet.tcp.mss_ifmtu	Integer	ja
net.inet.tcp.mssdflt	Integer	ja
net.inet.tcp.recvspace	Integer	ja
net.inet.tcp.rfc1323	Integer	ja
net.inet.tcp.rstppslimit	Integer	ja

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernel-Variable	Typ	änderbar?
net.inet.tcp.sack	Integer	ja
net.inet.tcp.sendspace	Integer	ja
net.inet.tcp.slowhz	Integer	nein
net.inet.tcp.syn_bucket_limit	Integer	ja
net.inet.tcp.syn_cache_interval	Integer	ja
net.inet.tcp.syn_cache_limit	Integer	ja
net.inet.tcp.timestamps	Integer	ja
net.inet.tcp.win_scale	Integer	ja
net.inet.udp.checksum	Integer	ja
net.inet.udp.recvspace	Integer	ja
net.inet.udp.sendspace	Integer	ja
net.inet6.icmp6.errppslimit	Integer	ja
net.inet6.icmp6.mtudisc_hiwat	Integer	ja
net.inet6.icmp6.mtudisc_lowat	Integer	ja
net.inet6.icmp6.nd6_debug	Integer	ja
net.inet6.icmp6.nd6_delay	Integer	ja
net.inet6.icmp6.nd6_maxnudhint	Integer	ja
net.inet6.icmp6.nd6_mmaxtries	Integer	ja
net.inet6.icmp6.nd6_prune	Integer	ja
net.inet6.icmp6.nd6_umaxtries	Integer	ja
net.inet6.icmp6.nd6_useloopback	Integer	ja
net.inet6.icmp6.nodeinfo	Integer	ja
net.inet6.icmp6.rediraccept	Integer	ja
net.inet6.icmp6.redirtimeout	Integer	ja
net.inet6.ip6.accept_rtadv	Integer	ja
net.inet6.ip6.anonportmax	Integer	ja
net.inet6.ip6.anonportmin	Integer	ja

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernel-Variable	Typ	änderbar?
net.inet6.ip6.auto_flowlabel	Integer	ja
net.inet6.ip6.v6only	Integer	ja
net.inet6.ip6.dad_count	Integer	ja
net.inet6.ip6.defmcasthlim	Integer	ja
net.inet6.ip6.forwarding	Integer	ja
net.inet6.ip6.gif_hlim	Integer	ja
net.inet6.ip6.hdrnestlimit	Integer	ja
net.inet6.ip6.hlim	Integer	ja
net.inet6.ip6.kame_version	String	nein
net.inet6.ip6.keepfaith	Integer	ja
net.inet6.ip6.log_interval	Integer	ja
net.inet6.ip6.lowportmax	Integer	ja
net.inet6.ip6.lowportmin	Integer	ja
net.inet6.ip6.maxfragpackets	Integer	ja
net.inet6.ip6.redirect	Integer	ja
net.inet6.ip6.rr_prune	Integer	ja
net.inet6.ip6.use_deprecated	Integer	ja
net.inet6.ipsec6.ah_net_deflev	Integer	ja
net.inet6.ipsec6.ah_trans_deflev	Integer	ja
net.inet6.ipsec6.def_policy	Integer	ja
net.inet6.ipsec6.ecn	Integer	ja
net.inet6.ipsec6.esp_net_deflev	Integer	ja
net.inet6.ipsec6.esp_trans_deflev	Integer	ja
net.inet6.ipsec6.inbound_call_ike	Integer	ja
net.inet6.udp6.recvspace	Integer	ja
net.inet6.udp6.sendspace	Integer	ja
net.key.acq_exp_int	Integer	ja

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernel-Variable	Typ	Änderbar?
net.key.acq_maxtime	Integer	ja
net.key.ah_keymin	Integer	ja
net.key.debug	Integer	ja
net.key.esp_auth	Integer	ja
net.key.esp_keymin	Integer	ja
net.key.kill_int	Integer	ja
net.key.random_int	Integer	ja
net.key.spi_max_value	Integer	ja
net.key.spi_min_value	Integer	ja
net.key.spi_try	Integer	ja
Prozesse		
proc.<pid>.corename	String	ja
proc.<pid>.rlimit.coredumpsize.hard	Integer	ja
proc.<pid>.rlimit.coredumpsize.soft	Integer	ja
proc.<pid>.rlimit.cputime.hardi	nteger	ja
proc.<pid>.rlimit.cputime.soft	Integer	ja
proc.<pid>.rlimit.datasize.hard	Integer	ja
proc.<pid>.rlimit.datasize.soft	Integer	ja
proc.<pid>.rlimit.filesize.hard	Integer	ja
proc.<pid>.rlimit.filesize.soft	Integer	ja
proc.<pid>.rlimit.maxproc.hard	Integer	ja
proc.<pid>.rlimit.maxproc.soft	Integer	ja
proc.<pid>.rlimit.memorylocked.hard	Integer	ja
proc.<pid>.rlimit.memorylocked.soft	Integer	ja
proc.<pid>.rlimit.memoryuse.hard	Integer	ja
proc.<pid>.rlimit.memoryuse.soft	Integer	ja
proc.<pid>.rlimit.stacksize.hard	Integer	ja

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernel-Variable	Typ	änderbar?
proc.<pid>.rlimit.stacksize.soft	Integer	ja
User-Space		
user.bc_base_max	Integer	nein
user.bc_dim_max	Integer	nein
user.bc_scale_max	Integer	nein
user.bc_string_max	Integer	nein
user.coll_weights_max	Integer	nein
user.cs_path	String	nein
user.expr_nest_max	Integer	nein
user.line_max	Integer	nein
user.posix2_c_bind	Integer	nein
user.posix2_c_dev	Integer	nein
user.posix2_char_term	Integer	nein
user.posix2_fort_dev	Integer	nein
user.posix2_fort_run	Integer	nein
user.posix2_localedef	Integer	nein
user.posix2_sw_dev	Integer	nein
user.posix2_upe	Integer	nein
user.posix2_version	Integer	nein
user.re_dup_max	Integer	nein
Herstellerspezifisches		
vendor.<vendor>.*	?	herstellerspezifisch
Virtual File System		
vfs.generic.usermount	Integer	ja
vfs.nfs.iothreads	Integer	ja

Tabelle 4.13 (Fortsetzung auf nächster Seite)

Name der Kernel-Variable	Typ	änderbar?
Virtuelle Maschine		
vm.anonmax	Integer	ja
vm.anonmin	Integer	ja
vm.execmax	Integer	ja
vm.execmin	Integer	ja
vm.filemax	Iteger	ja
vm.filemin	Integer	ja
vm.loadavg	Struktur	nein
vm.nkmempages	Integer	nein
vm.uvmexp	Struktur	nein
vm.uvmexp2	Struktur	nein
vm.vmmeter	Struktur	nein
Linux-Emulation		
emul.linux.kern.ostype	String	nein
emul.linux.kern.osrelease	String	nein
emul.linux.kern.osversion	String	nein

Tabelle 4.13: Die wichtigsten NetBSD-Kernelvariablen

Die hier aufgelisteten Kernel-Variablen korrespondieren direkt mit den Kernel-Werten, die allerdings in anderen Schreibweise angezeigt werden. Die Unterschiede listen die Manpages *sysctl (3)* und *sysctl (8)* auf.

Es gibt weitere Kernel-Variablen wie *kern.rootdevice*, die in dieser Liste nicht aufgeführt sind und nicht der Modifikation dienen. Sie erhalten die komplette Liste mit allen momentan eingetragenen Werte über einen Aufruf von *sysctrl -a | more*.

5 Die Package-Sammlung

von Markus Illenseer

Im Internet findet man sehr viel Software für unix-ähnliche Systeme. In der Regel kann man diese Software problemlos auch unter NetBSD einsetzen, manchmal sind dazu aber mehr oder wenige Veränderungen notwendig, etwa eine Anpassung der Pfade, der Systemaufrufe oder ähnliches. Die NetBSD-Package-Sammlung versucht, solche notwendigen Änderungen zu integrieren und die Installation der damit lauffähigen Software zu vereinheitlichen sowie auch eine De- oder Re-Installation zu ermöglichen. Dazu werden zwei Grundsysteme angeboten: einmal die Installation direkt aus den Quellen und die vereinfachte Installation aus speziellen Binärarchiven, in denen die Software bereits für NetBSD vorkonfektioniert ist.

Vorab eine wichtige Konvention, damit keine Verständnisprobleme entstehen: Unter »Packages« wird eine Sammlung von Programmen, Programmbibliotheken, Tools, grafischen Oberflächen und allem, was das Herz des Benutzers und Administrators begehrt, verstanden. Ein einzelnes Package ist eigentlich ein Archiv mit automatischer Installationsanleitung für eine Software. Beispielsweise gibt es ein Package für das Programm »lynx«, einen textbasierten Web-Browser. Das Package umfaßt das ausführbare Programm, die Manpage, Konfigurationsdateien und weitere Dateien. Nach der Installation sind das Programm und alle weiterhin installierten Dateien in einer Systemdatenbank registriert und stehen jedem Benutzer zur Verfügung.

Die eigentlichen Packages sind nicht Teil des NetBSD-Betriebssystems und müssen in der Regel gesondert erworben werden. Bei vielen NetBSD-Distributionen sind allerdings sofort installierbare Packages enthalten, so daß die wichtigsten Programme schnell installiert sind.

Das NetBSD-Package-System ist Teil des NetBSD-Betriebssystems und verarbeitet und erzeugt oder löscht die verfügbaren Packages. Die Betriebssystemprogramme *pkg_add*, *pkg_delete* und *pkg_info* übernehmen den größten Teil der Installation von Binär-Packages und erzeugen den Package-Bereich, falls er noch nicht vorhanden ist.

Packages (engl. »Paket«) sind eine Sammlung von Dateien und Anweisungen für die Erzeugung von Dritt-Software mit Hilfe des NetBSD-Package-Systems. Die Packages werden traditionell in »/usr/pkgsrc« installiert. Auch wenn ein Package also nur aus Anweisungen besteht, stellt es insgesamt ein Software-Paket dar, da die Dritt-Software als Quelltext aus dem Internet nachgeladen, an NetBSD angepaßt und installierfähig gebaut wird. Ein Package enthält keine lauffähigen Programme.

Binär-Packages sind Archive mit lauffähigen Programmen eines Packages. Binär-Packages können sofort installiert werden und sind in der Regel sofort lauffähig. Ein Binär-Package wird aus einem Package optional generiert. Binär-Packages werden in der Regel unter »/usr/pkgsrc/packages« abgelegt. Installiert werden die im Binär-Package enthaltenen, sofort lauffähigen Programme per Voreinstellung nach »/usr/pkg«.

Die Paketdatenbank

Wird ein Programm über das Paketsystem installiert, legt das Paketverwaltungsprogramm automatisch einen Eintrag in der Paketdatenbank an. Diese Paketdatenbank ist nichts anderes als eine Liste von Verzeichnissen in »/var/db/pkg«. Jedes der dort befindlichen Verzeichnisse besitzt den Namen inklusive der Versionsnummer des zugehörigen installierten Software-Pakets. In diesem Verzeichnis befinden sich wiederum mehrere Dateien, die von den Verwaltungstools ausgewertet werden und Aufschluß über das Softwarepaket geben. Alle Dateinamen beginnen mit einem Plus-Zeichen und sind in Großbuchstaben abgelegt. Sie werden beim Erzeugen eines Pakets im Archiv abgelegt:

+BUILD_INFO	Liste der Parameter bei der Paketerzeugung.
+BUILD_VERSION	Pfade, Patches und weitere Build-Informationen.
+COMMENT	Die einzeilige Paketinformation.
+CONTENTS	Liste der installierten Dateien und angelegten Verzeichnisse.
+DESC	Die ausführliche Paketinformation.
+SIZE_ALL	Die Gesamtgröße der installieren Dateien des Pakets inklusive der abhängigen Dateien.
+SIZE_PKG	Größe des Pakets selbst (ohne die abhängige Dateien).

Außer diesen Pflichtdateien können weitere optionale Dateien abgelegt sein:

+REQUIRED_BY	Liste der Pakete, die dieses Paket als Abhängigkeit besitzen.
+DISPLAY	Informationen, die bei der Installation angezeigt werden.
+DEINSTALL	Deinstallationsskript, beispielsweise bei *sudo*.

Ein *Distfile* (»Distributed File«, englisch für »verteilte Dateien«) ist ein Archiv der originalen, unveränderten Dritt-Software in Form von Quelltext. Ein Package erzeugt mit Hilfe der Anweisungen aus dem Distfile ein installierbares Paket der Dritt-Software. Da die Distfiles aus dem Internet geladen werden – nicht gezwungenermaßen zentralisiert vom NetBSD-Server, sondern zumeist von den originalen Homepages – speichert das Package-System diese Archiv in »/usr/pkgsrc/distfiles« ab, damit bei einer erneuten Installation keine Bandbreite vergeudet wird.

Der Vollständigkeit halber sei erwähnt, daß der Begriff »Port« die verwendete Terminologie unter FreeBSD und OpenBSD für Packages ist. NetBSD versteht unter »Port« eine weitere unterstützte CPU-Architektur für das Betriebssystem selber.

Ein Package ist also eine Zusammenstellung von Dateien, die die originale Software im Quelltext nachladen kann, anpaßt und auf Wunsch installiert. Unix-Software wird normalerweise als Quelltext zur Verfügung gestellt und unter BSD dann durch das Package-System angepaßt (»Patch«), kompiliert und installiert. Die erforderliche Zusammenstellung und die Anpassungen werden von Mitgliedern der NetBSD-Gruppe, aber auch von engagierten anderen Personen durchgeführt. Im zweiten Teil dieses Kapitels erfahren Sie, wie Sie Packages für NetBSD selbst erzeugen.

Quelltextpakete oder Binär-Packages

Man kann darüber streiten, was besser ist: die Installation aus den Quellen oder aus einem vorkompilierten Paket. Das zweite ist immer schneller, gerade bei nicht aktueller Hardware kann die Übersetzung eines Pakets Tage dauern; bereits auf einem AMD K6/2 400 dauert das Kompilieren von KDE 3.0.3 mehrere Tage. Man ist beim Einspielen von Binärpaketen aber nie vor Überraschungen gefeit. Es reicht schon, daß eine wichtige Bibliothek, die unbedingt benötigt wird, in einer zu alten oder zu neuen Version vorliegt, um das Einspielen zum Abenteuer werden zu lassen. Dabei kann es dann schnell passieren, daß das aus dem Internet gezogene Paket instabil ist.

Wer mehrere verschiedene BSD-Maschinen (entweder verschiedener Architektur oder auch mit verschiedenen BSD-Betriebssystemen ausgestattet) besitzt, sollte die Installation aus den Quelltexten aber unbedingt in Erwägung ziehen. Obwohl die Quellpakete in der Regel größer als ein einzelnes Binärpackage sind, spart man sich doch beim einmaligen Laden der Quelltexte gegenüber dem mehrfachen Download der Binärpakete viel Download-Zeit. Während des Kompilierens des Packages kann außerdem auf anderen Konsolen normal mit dem System weitergearbeitet werden.

Alternativ bietet NetBSD auch bereits vorgefertigte und sofort installierbare Packages in binärer Form an. Sie sind ohne Kompilierung sofort nutzbar. Diese Form der Distribution ist in den meisten Fällen ausreichend und angenehm bequem. Dafür ist man dann darauf angewiesen, darauf zu warten, bis wichtige Änderungen in einer originalen Software in einem Package integriert sind und eben zusätzlich auch vorkompiliert und in ausführbarer Form als Binär-Package archiviert werden. Einige Software-Pakete dürfen lizenzrechtlich nicht in solchen Paketen angeboten werden. Dann ist man gezwungen, das Package-System im Quelltext zu verwenden.

Andere unix-ähnliche Betriebssysteme bieten ähnliche Distributionen für zusätzliche Software an. Unter Solaris werden Packages zur Verfügung gestellt, unter Linux (je nach Distributor unterschiedlich) heißen sie meist RPM-Archive. Unter FreeBSD und OpenBSD sind sie als Ports bekannt. Ihnen allen gemein sind eine Datenbank und eine kontinuierliche Verfolgung der Abhängigkeiten, der installierten Programmdateien, Manpages und Konfigurationsdateien.

Übrigens ist die NetBSD-Package-Kollektion so gut gelungen, daß andere Systeme sie zu großen Teilen übernehmen. So wird für das aktuelle MacOS X von Apple eine soge-

nannte »Darwin Package Collection« angeboten, die auf dem Package-System von NetBSD basiert. Für Solaris von Sun gibt es das System »Zoularis«. Es wird sogar von Solaris-Entwicklern selbst unterstützt. Dennoch hat das NetBSD-Package-System seinen Ursprung im Ports-System von FreeBSD.

Die Software, die mit dem Package-System zur Verfügung gestellt wird, ist kein Bestandteil von NetBSD und unterliegt den individuellen Lizenzbestimmungen der einzelnen Pakete. Der Großteil dieser Software ist unter der GPL frei verteilbar. Manche Lizenz erlaubt aber zum Beispiel nicht, daß man sie an einem anderen Ort als dem ursprünglichen zur Verfügung stellt. Das NetBSD-Package-System stellt für diese Art aber dennoch eine Installations- und Download-Möglichkeit zur Verfügung, damit der Administrator des Systems nicht selbst lange suchen muß.

Werden Packages installiert, merkt sich das System, welche Dateien eingebunden wurden. Bei einer Deinstallation wird diese Dateiliste herangezogen, damit alle relevanten Dateien gelöscht werden können. Eine Ausnahme bilden wichtige Konfigurationsdateien – sie werden nicht gelöscht, da sie ja bei einem Upgrade auf eine neue Version wichtig sein können. Es wird aber bei der Deinstallation darauf hingewiesen, daß man diese Dateien bei Bedarf manuell löschen sollte. Damit bleibt ein Datei-Chaos aus und vor allem werden Karteileichen verhindert. Auch wird vor einer Installation geprüft, ob bestimmte andere Packages bereits installiert sind. Diese Abhängigkeitsprüfung ist zum Beispiel wichtig, wenn ein Grafiktool eine Grafik-Bibliothek erfordert, die noch nicht installiert ist. Das Package-System gibt dann genau an, welche Bibliothek vorab zu installieren ist. Je nachdem, wie man die Packages einspielt, wird die Installation der erforderlichen Packages selbsttätig durchgeführt. Wichtig ist also, ausschließlich die dem Package-System beigefügten Programme zur Installation und Deinstallation zu verwenden, da sie vor dem Löschen prüfen, ob das zu löschende Package nicht von anderen Packages benötigt wird.

Eine Mischung mit manuell installierten Software-Paketen ist möglich, etwa weil sie zu neu sind, um in der NetBSD-Package-Sammlung enthalten zu sein. Die Abhängigkeitsprüfung verwendet die normalen System-Pfade zum Finden bereits installierter Bibliotheken. Praktischer ist die Verwendung der NetBSD-Packages zur Installation der Abhängigkeiten aber dennoch. Das Mischen ist außerdem nicht ganz einfach.

5.1 Installation und Wartung auf der Kommandozeile

Es gibt mehrere Wege der Installation von Packages. Der bequemste und mit Abstand schnellste ist die Nutzung von *pkg_add* zur Installation von Binär-Distributionen wie sie beispielsweise auf CD-ROMs oder natürlich im Internet zu finden sind. Der sicherste und deutlich umfangreicher konfigurierbare Weg ist die Verwendung der Package-Sourcen.

Dennoch gelten für beide Wege die folgenden Regeln, damit die Installation gelingt:

Für die Installation der meisten Packages werden Administratorrechte (Benutzer *root*) benötigt. Theoretisch würde es ausreichen, nach einer ersten Installation den zukünftigen Dateibaum unterhalb »/usr/pkg« und »/usr/pkgsrc« zum Beispiel der Gruppe *wheel* zugehörig zu setzen und einem Benutzer dann in diese Gruppe einzufügen. Dieser Be-

nutzer kann dann große Teile der Packages auch ohne *root*-Rechte installieren. Alternativ reicht es, nur bei der Installation die *root*-Rechte zu nehmen, indem das Kommando *su* vorangestellt wird, bei der Kompilierung aber einen normalen User einzusetzen. Konzepte für das Benutzen von *root*-Rechten für normale Benutzer können auch mit Hilfe des Package *sudo* erarbeitet werden. Bei Daemons wie einem FTP- oder httpd-Server ist es aber unumgänglich, *root*-Rechte für die Installation zu besitzen.

Verfügbare Binär-Packages finden Sie unter der Hauptadresse *ftp://ftp.netbsd.org/pub/NetBSD/packages/<release>/<arch>/All/*, wobei *<release>* die NetBSD-Version (zum Beispiel 1.6.1), und *<arch>* die Hardware-Architektur ist (zum Beispiel »i386« oder »amiga«). Nicht immer gibt es alle Verzeichnisse in einer höheren Version von NetBSD. Das Verzeichnis 1.6 ist also eine Basis. Aktuell neu zur Verfügung gestellte Packages werden dann unter dem Verzeichnis 1.6.1 oder 1.6.2 abgelegt. Sollten Sie bei der Versionsnummer und Architektur unsicher sein, rufen Sie das Programm *uname* auf. *uname -rp* zeigt die Version und Architektur an:

```
root@desktop: ~#uname -rp
1.6 i386
```

Eine ausführliche Liste der verfügbaren Packages finden Sie (laufend aktualisiert) unter *ftp://ftp.netbsd.org/pub/NetBSD/packages/pkgsrc/README.html.* Alternative und brauchbare Mirrors sind im deutschsprachigen Raum unter den URLs *ftp://ftp.de.netbsd.org* und *ftp://ftp.de2.netbsd.org* zu finden.

5.1.1 Schnellinstallation über Binärpakete

Am schnellsten wird eine gewünschte Software mit dem Befehl

```
%> pkg_add <URL-zum-Binär-Package>
```

direkt aus dem Internet installiert:

```
%> pkg_add -v ftp://ftp.de.netbsd.org/pub/NetBSD/packages/1.6/i386/All/↵
lynx-2.8.4.0b19.tgz
```

Unter der Annahme, daß eine CD-ROM mit Binär-Packages unter »/mnt/cdrom« eingebunden ist, funktioniert die Installation von dort genauso:

```
%> pkg_add -v /mnt/cdrom/packages/1.6/i386/All/lynx-2.8.4.0b19.tgz
```

Dieser Aufruf installiert den textbasierten Web-Browser »lynx«. Die Option *-v* zeigt dabei die Installationsschritte an, damit man einen Eindruck erhält, welche Arbeitsschritte *pkg_add* durchführt. Denken Sie dabei daran, daß Sie *root*-Rechte besitzen müssen beziehungsweise als User *root* eingeloggt sind. Da die Pfade zum Teil recht lang werden und es recht mühsam ist, diese Pfade immer wieder neu eingeben zu müssen,

kann der Pfad oder der URL in der Umgebungsvariablen *PKG_PATH* gesichert werden. Bei Verwendung der Bash als System-Shell wird dazu der folgende Aufruf getätigt:

```
#export PKG_PATH=ftp://ftp.de.netbsd.org/pub/NetBSD/packages/1.6/i386/All/
```

wenn die Pakete direkt aus dem Internet bezogen werden, oder

```
#export PKG_PATH=/mnt/cdrom/packages/1.6/i386/All/
```

wenn die Pakete lokal von der CD-ROM installiert werden. Bei Verwendung der *csh* als System-Shell sind die Aufrufe etwas anders:

```
%> setenv PKG_PATH ftp://ftp.de.netbsd.org/pub/NetBSD/packages/1.6/i386/All/
```

oder

```
%> setenv PKG_PATH /mnt/cdrom/packages/1.6/i386/All/
```

Danach reicht die folgende Kurzform für die Installation des Pakets:

```
%> pgk_add -v lynx-2.8.4.0b19.tgz
```

Jetzt ein anderes Beispiel:

```
#pkg_add -v ftp://ftp.de.netbsd.org/pub/NetBSD/packages/1.6/i386/All/xview
```

Oder, nach dem Setzen der Variable *PKG_PATH*:

```
%> pkg_add -v xview
```

Sie sehen, daß hier kein expliziter Dateiname angegeben ist. Die Datei *xview* gibt es auch nicht. Es gibt aber folgende Dateien auf dem FTP-Server, die mit *xview* anfangen (zum Zeitpunkt der Drucklegung dieses Buchs):

```
xview-3.2.5.tgz
xview-all-3.2.5.tgz
xview-clients-3.2.5.tgz
xview-config-3.2.5.tgz
xview-config-3.2.1nb5.tgz
xview-config-3.2.1nb2.tgz
xview-lib-3.2.5.tgz
xview-lib-3.2.1nb5.tgz
xview-lib-3.2.1nb2.tgz
```

pkg_add installiert die erste gefundene Datei – wobei das hier per Zufall die richtige ist; jede andere aus der obigen Liste würde aber auch funktionieren, aber wir wollten das eigentliche *xview* installieren. Dabei stellt *pkg_add* aber Abhängigkeiten der folgenden weiteren Packages fest:

```
xview-clients>=3.2.1
xview-config>=3.2.1
xview-config>=3.2.1nb2
xview-lib>=3.2.1
```

Es muß also unter anderem mindestens noch ein *xview-client* mit der Versionsnummer größer als 3.2.1 vorliegen. Da dies nicht der Fall ist, lädt *pkg_add* von der URL das Pakkage »xview-clients-3.2.5.tgz« nach, installiert es (und stellt dabei eventuell andere Abhängigkeiten fest ...).

Das bedeutet, daß der Installationsvorgang notfalls rekursiv weitere Packages nachlädt und installiert, und zwar solange, bis alle Bedingungen erfüllt sind, um das ursprünglich gewünschte Package lauffähig zu installieren. Unter Umständen zieht dies eine Installations-Orgie nach sich!

Ganz vermeiden kann man solche »Orgien« nicht, denn zu einem lauffähigen System gehört oft mehr als nur das originale Package. Auch haben sich die Zusammensteller der Packages eigene Gedanken bei der Abhängigkeit gemacht, um später eine Installation im Zweifel lieber übervollständig zu haben, als daß eine kleine Komponente fehlt und später nachinstalliert werden muß. Ein Beispiel für übergroße Vorsicht ist die automatische Installation von Zeichensätzen bei der Installation von Ghostscript, einem Anzeigeprogramm für Postscript- und PDF-Dateien. Ohne die Zeichensätze ist Ghostscript nahezu unbrauchbar. Was aber, wenn nur ein oder zwei Zeichensätze ausreichten? Das Package-System läßt leider keine Wahl zu. Dazu müssen Sie das Package-System im Sourcecode installieren und das entsprechende Makefile editieren.

5.1.2 Das Package-Infosystem

Nach der erfolgreichen Installation eines Softwarepakets kann mit dem Befehl *pkg_info* mehr über das Paket (hier beispielsweise *xview*) herausgefunden werden:

```
%> pkg_info xview
```

Ein Überblick über alle installierten Packages liefert ein einfaches *pkg_info*, dessen Ausgabe zum Beispiel nach *more* umgeleitet werden kann: *pkg_info | more* oder zur Suche nach einem nur ungefähr bekannten Namen eines Packages *pkg_info | grep -i xv*.

5.1.3 Packages löschen

Wenn Sie einen Fehler gemacht haben und Sie sich entscheiden, daß Sie das Paket gar nicht auf Ihrem System haben möchten, wenden Sie *pkg_delete* an.

```
%> pkg_delete xview
```

Dieser Aufruf deinstalliert allerdings nur das Basis-Paket. Die weiteren Pakete *xview-clients* und Co werden nicht deinstalliert – sie könnten ja noch von anderen Programmen benötigt werden! Will man die anderen Pakete dennoch deinstallieren, geht das mit der Option *-r*:

```
%> pkg_delete -r xview
```

Auch kann man anders vorgehen und ein Package deinstallieren und alle Packages mit deinstallieren, von denen das zu löschende ursprünglich abhängig war. Dazu braucht man die Option *-R*.

5.1.4 Die Paket-Hierarchie

Doch wo hat *pkg_add* die Pakete installiert? Es wird eine neue Hierarchie namens »/usr/pkg« angelegt. Erfahrene Administratoren erkennen sofort eine neue, Ihnen zumeist unbekannte Hierarchie. Um Unstimmigkeiten mit Nicht-NetBSD-Packages aus dem Wege zu gehen, hat man eine neue Hierarchie eingeführt. Sie basiert nicht auf »/usr/local« oder »/opt« und installiert alle Dateien ausnahmslos in »/usr/pkg«, auch die Konfigurationsdateien und Bibliotheken. Eine Ausnahme gibt es allerdings: Packages für das X-System werden in »/usr/X11R6« installiert.

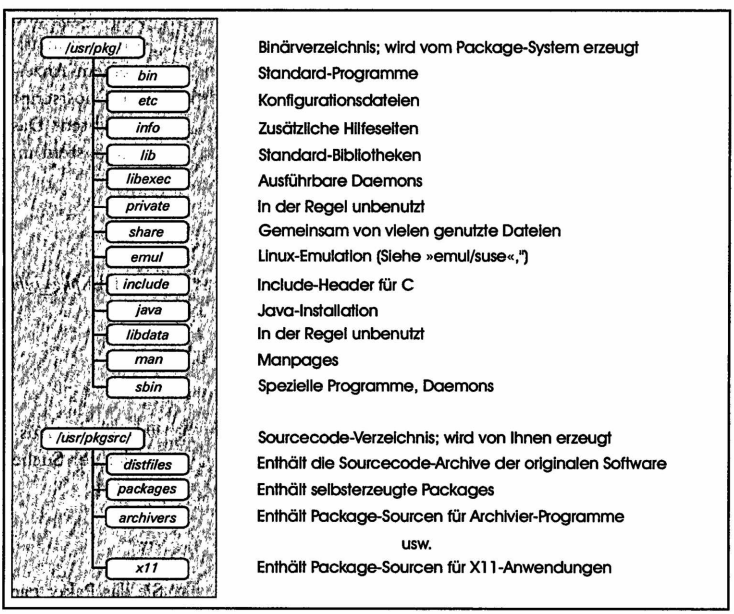


Abb. 5.1: Die Paket-Verzeichnishierarchie

Wenn Sie nicht die Installation mittels *pkg_add* verwenden, sondern die Installation per Sourcecode aus dem Package Sourcecode-Verzeichnis vorziehen, können Sie diesen Zielpfad verändern und beispielsweise alles nach »/opt« installieren lassen.

Weil »/usr/pkg« zumeist in der Standard-Partition »/usr« beheimatet ist, sollte der Platz ausreichend für die Installation der Packages sein. Jedes Package wird dort unkomprimiert installiert. Achten Sie bei der Installation weiterer Packages auf ausreichend Plattenplatz. Das Package-System berücksichtigt allerdings den freien Platz, bevor es installiert.

5.2 Installation mittels Package-Sourcetree

Noch mehr Plattenplatz und vor allem Systemresourcen werden benötigt, installiert man die Packages aus den Sourcen. Dafür installiert man einen initialen Sourcetree – eine Hierarchie mit Sourcen – der dann pro verfügbarem Package ein Verzeichnis enthält. In diesem Verzeichnis befinden sich, wie bei Quellen üblich, Makefiles, die dann dafür sorgen, daß das eigentliche Package in Source-Form vom Internet geladen, ausgepackt, angepaßt, kompiliert und installiert wird. Weiterhin können diese Makefiles aber auch Packages deinstallieren (*deinstall*), erneuern (*update*) und Informationen über Pakkages ausgeben.

Der initiale Package-Sourcetree muß einmal vom Internet geladen und ausgepackt werden. Danach kann man diesen Sourcetree mit dem Programm *cvs* nach Bedarf aktualisieren. Damit ist man stets auf dem aktuellsten Stand beziehungsweise kann diesen nach Belieben erreichen. Dies ist besonders wichtig, wenn neue Programmversionen erscheinen, aber Abhängigkeiten von anderen Programmen besitzen. Für die Aktualisierung ist eine Internet-Verbindung notwendig.

Das Archiv des initialen Package-Sourcetrees wird wöchentlich aktualisiert und unter der URL *ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-current/tar_files/pkgsrc.tar.gz* zur Verfügung gestellt. Achtung, das Archiv ist derzeit um die zehn Mbyte groß, ausgepackt benötigt es etwa 90 Mbyte Platz! Nachdem Sie einmal das Archiv installiert haben, aktualisieren Sie dieses Archiv mit cvs – die Veränderungen sind in der Regel dann nur noch wenige Kbyte groß. Laden Sie das Archiv und packen Sie es nach »/usr/« aus:

```
%> cd /tmp
%> ftp ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-current/tar_files/pkgsrc.\
tar.gz
%> cd /usr
%> tar zvxf /tmp/pkgsrc.tar.gz
```

Nach dem Auspacken kann das originale Archiv bei Bedarf gelöscht werden:

```
%> rm /tmp/pkgsrc.tar.gz
```

Befehl	Funktion
fetch	Holt das Paket aus dem Internet.
fetch-list	Erzeugt eine Liste der benötigten Dateien für das Paket inklusive lokal fehlender abhängiger Dateien. Die Ausgabe erfolgt auf den Bildschirm und sollte in eine Datei umgeleitet werden.
checksum	Prüft die Checksummen der lokal vorhandenen Distfiles.
extract	Entpackt das Orginal-Distributionsarchiv in das Unterverzeichnis »work«. Dabei wird meistens ein *tar*-Aufruf in »work« ausgeführt. Damit befindet sich in »work« ein weiteres Unterverzeichnis, das so heißt wie es im Original-Archiv als Unterverzeichnis vorgesehen ist.
configure	Führt den Lauf von *GNU configure* durch.
build	Kompiliert das Paket (entspricht dem einfachen Aufruf von *make*).
install	Installiert das Programmpaket ins System.
package	Erzeugt aus dem installierten Programmpaket ein Binärpaket. Dabei müssen die Temporärdateien im Arbeitsunterverzeichnis »work« vorhanden sein. Ist das nicht der Fall, muß für das Packetieren das Programm *pkg_tarup* aus den Packages verwendet werden.
makesum	Erzeugen einer neuen Checksummendatei. Dieser Befehl sollte nur eingesetzt werden, wenn ein neues Verzeichnis gebaut oder eine Datei umgepackt wurde (beispielsweise wenn eine vorhandene .tar.gz-Datei für die Verwendung unter NetBSD nach ».tar.bz2« umgepackt wurde). Nur lokal verwendbar.
clean	Löscht die Interimsdateien aus dem Verzeichnis.
clean-depends	Reinigt abhängige Verzeichnissen. Nur lokal verwendbar.
reinstall	Installiert das Paket erneut ohne Überprüfung, ob es bereits installiert ist (entspricht *make NO_PKG_REGISTER=Yes install*).
update	Deinstalliert zuerst das Paket und alle abhängigen Pakete und führt dann eine Neuinstallation durch.
deinstall	Löscht das Paket aus dem System (entspricht *pkg_delete*).
clean	Reinigen des aktuellen PkgSrc-Verzeichnisses. Dieser Befehl ist auch global und in Unterverzeichnissen wie »www« oder »x11« einsetzbar.
readme	Anlegen einer Readme-Datei »README.html« mit den Informationen zum Programmpaket. Überall möglich.
info	Zeigt die Information zum Paket an. Dieser Aufruf entspricht *pkg_info*.

Tabelle 5.1: Die wichtigsten Befehle von make in Package-Verzeichnis

5.2.1 Installation von Packages

Der Package-Sourcetree ist in verschiedene Kategorien unterteilt, in denen dann die einzelnen Packages zu finden sind. Das oben angesprochene »lynx« ist zum Beispiel in der Hierarchie »/usr/pkgsrc/www/lynx« zu finden.

Um ein Package auf dem System zu installieren, wechseln Sie in das entsprechende Verzeichnis und starten die Kompilierung:

```
%> cd /usr/pkgsrc/www/lynx
%> make
```

Dies lädt den Source von lynx im Internet von der Homepage von lynx. Im Beispiel ist der URL *http://lynx.isc.org/release/* in der Datei »Makefile« hinterlegt. Sollte dieser URL nicht verfügbar sein, kennt das Makefile noch weitere Adressen, unter denen es lynx im Sourcecode finden kann. Dies gilt nicht für alle Packages, manche laden beispielsweise aufgrund spezieller Lizenzvorschriften nur von einem ganz bestimmten URL.

Als nächstes prüft *make*, ob noch Patches zum aktuellen Archiv vom *lynx* einzuspielen sind und führt diesen Vorgang gegebenenfalls durch. Dabei legt es ein neues Verzeichnis »work« im entsprechenden Package-Verzeichnis an, in dem sich die Sourcen des Packages befinden, und ändert die Dateien nach Patch-Anweisung.

Danach prüft *make*, ob es Abhängigkeiten gibt. Falls ja, also beispielsweise andere Programme zur vollständigen Funktion erforderlich sind, wechselt *make* in das entsprechende Package und fängt dort mit der Kompilierung (und mit dem Laden des Quelltexts vom Internet) an.

Nach der erfolgreichen Kompilierung – die je nach System und Package natürlich unterschiedlich lange dauert – werden das Package und alle Abhängigkeiten installiert:

```
%> make install
```

Spätestens dafür benötigt man in der Regel *root*-Rechte.

Damit die Quellen der installierten Programme bei Abfragen oder durch abhängige andere Packages nicht ständig aus dem Internet nachgeladen werden müssen, speichert das Package-System erfolgreich geladene Sourcen von Packages in »/usr/pkgsrc/distfiles« in der dort ebenfalls vorhandenen Package-Source-Hierarchie ab.

Packages können Sie nun wahlweise mit *make deinstall* im entsprechenden Package-Verzeichnis oder mit Hilfe von *pkg_delete* löschen. Die Abhängigkeiten werden dabei nicht berücksichtigt, es sei denn, das zu deinstallierende Programm selber ist eine Notwendigkeit für andere Packages. Dann kann es nicht gelöscht werden, außer man ist sich seiner Sache sicher. Dann kann die Option *-f* für »force« zur Anwendung kommen.

Informationen über alle installierten Packages erhalten Sie auch nach der Installation mittels *make install* wie gewohnt mit *pkg_info*.

Befehl	Erfolgsdatei in work/	Funktion
fetch	Keine. Es wird anhand des Vorhandenseins der Datei geprüft.	Laden des Distfiles, falls es nicht bereits in »/usr/pkgsrc/distfiles« vorhanden ist.
checksum	Keine. Es wird bei jedem Aufruf erneut getestet.	Überprüfung der Checksumme des oder der Distfiles. Es wird die entsprechende Erfolgsmeldung angezeigt.
extract	.extract_done	Das Quellarchiv wird entpackt.
patch	.patch_done	Einspielen der Patches (rekursiver Build der abhängigen Pakete erfolgt).
configure	.configure_done	Es wird der Lauf von *./configure* für das Softwarepaket durchgeführt.
(build)	.build_done	Kompilieren des Pakets. Der Parameter kann weggelassen werden. Diese Phase entspricht dem Aufruf von *make* ohne Parameter.
install	.install_done	Installation des erzeugten Pakets aus den Dateien nach dem Build-Lauf.
package	.package_done	Erzeugen des Binärpakets.

Tabelle 5.2: Die wichtigsten Parameter von make im Arbeitsverzeichnis zum Einbinden einer Software

Variable	Funktion
NO_CHECKSUM	Keine Checksummenprüfung auf die Quelldatei(en) durchführen.
NO_PKG_REGISTER	Keine Installationsprüfung durchführen und ein bereits installiertes Paket rücksichtslos überinstallieren.

Tabelle 5.3: Die wichtigsten Steuervariablen für make. Sie werden mit »=YES« eingeschaltet und sind sehr behutsam einzusetzen

5.2.2 Aufräumarbeiten

Um Platz zu sparen, können Sie nach erfolgreicher Installation eines Packages im entsprechenden Verzeichnis die temporären Dateien und das work-Verzeichnis löschen, beispielsweise mit

```
%> cd /usr/pkgsrc/www/lynx
%> make clean
```

Ein *make clean-depends* löscht auch alle temporären Dateien, die bei der Installation der Abhängigkeiten entstanden.

Sie können optional auch regelmäßig ein *make clean* im Hauptverzeichnis von »pkgsrc« durchführen. Das löscht alle ausgepackten Dateien, temporären Dateien und Objektdateien, jedoch nicht die Sourcedateien im Verzeichnis »distfiles«.

5.2.3 Installations-Optionen

Da *make install* per Voreinstellung die Pakete in »/usr/pkg« ablegt, dies aber nicht jedermanns Geschmack ist, kann das Zielverzeichnis auch verändert werden. Setzen Sie dazu die Variable *LOCALBASE*:

```
%> export LOCALBASE=/usr/local
```

Damit erfolgen danach gestartete Installationen in */usr/local*. Bitte beachten Sie, daß Sie einen vernünftigen Ort für die Speicherung wählen sollten. »/usr« wäre überhaupt keine gute Idee und »/usr/pkgsrc«, »/opt« oder »/usr/src« beispielsweise würden mit großer Wahrscheinlichkeit Inkonsistenzen erzeugen.

Auch hier gilt: Packages für X11 werden dennoch in »/usr/X11R6/« installiert. Möchten Sie diese auch in einer anderen Hierarchie speichern, nutzen Sie die Variable *X11BASE* dazu.

Um Packages für X11 dennoch in *LOCALBASE* einzurichten, müssen Sie das Paket *xpkgwedge* (»pkgsrc/pkgtools/xpkgwedge«) installieren. Alle Variablen können auch dauerhaft in »/etc/mk.conf« gespeichert werden.

5.2.4 Eigene Binär-Packages erzeugen

So schön das Installieren mit Hilfe der originalen Sourcen der installierten Software ist, es hat vor allem den Nachteil, daß man auf Dauer viel Plattenplatz und Zeit benötigt. Eine Installation von Mozilla (Web-Browser) erfordert während des Kompilierens geschätzte 350 Mbyte. Das sprengt sogar in Zeiten günstiger Festplatten den Platz gehörig. Die Dauer des Übersetzungsvorgangs ist natürlich vom System abhängig, es werden aber wertvolle Ressourcen und Zeit benötigt, in der der Server schwer arbeitet.

Für sicherheitsbewußte Systemadministratoren ist die Installation von Software mittels Kompilierung der originalen Sourcen oft unumgänglich. Um nun aber nicht auf jedem betreuten Rechner diesen Vorgang durchführen zu müssen, können Sie auch eigene Package-Archive erzeugen:

```
%> cd /usr/pkgsrc/www/lynx
%> make package
```

Das erzeugte Package wird in »/usr/pkgsrc/packages/www/« abgelegt. Es kann dann im lokalen LAN auf den anderen Rechnern bequem zur Installation verwendet werden.

Wichtig zu wissen ist, daß manche Packages die Erzeugung oder Verteilung solcher Packages in binärer Form leider lizenzrechtlich nicht erlauben.

Binärpakete mit .tgz oder .tbz?

Zum Zeitpunkt, als dieses Buch geschrieben wurde, verwendete NetBSD als voreingestelltes Archivformat für Binär-Packages noch das zwar gängige, aber nicht besonders gut packende .tar.gz-Format (abgekürzt »tgz«); genauso läßt sich aber das inzwischen bei FreeBSD ebenfalls eingesetzte ».tar.bz2«-Format (abgekürzt »tbz«) einsetzen. Dazu muß nur in der globalen Steuerdatei »/etc/mk.conf« der Eintrag *PKG_SUFX=.tbz* angegeben werden. Sollte die Steuerdatei nicht vorhanden sein, muß sie neu angelegt werden. Das *tar*-Programm von NetBSD packt bzip2-gepackte Archive mit dem selben Parameter wie gzip-Archive aus. Auch bei *pkg_add* treten keine Kompatibilitätsprobleme auf. Auch das Mischen von »tgz«- und »tbz«-Archiven ist problemlos möglich. Das Verwaltungsprogramm *pkg_tarup* (»/usr/pkgsrc/pkgtools/pkg_tarup«), mit dem man aus installierten Programmen Binärpackages erzeugt, ignoriert die »mk.conf« und legt immer »tgz«-Dateien (im Verzeichnis »/tmp«) an. Man kann diese Dateien mit

```
gunzip archiv.tgz
bzip2 archiv.tar
mv archiv.tar.bz2 archiv.tbz
```

aber in das platzsparende Format umpacken.

Bei der Erzeugung des Binär-Packages wird das gerade eben durch *make install* erzeugte Package allerdings zur Konsistenzprüfung gelöscht. Installieren Sie das Paket erneut mit *pkg_add* und dem gerade neu erzeugten Binär-Package oder starten Sie *make install* erneut – es verwendet dann das erzeugte Binär-Package.

5.2.5 Ressourcen

Derzeit werden ungefähr folgende Ressourcen verwendet:

Architektur: 1.6/i386:

Distfiles	1500 Mbyte (NFS ok)
Kompletter Satz aller Programme	1000 Mbyte (NFS ok)
Temporärer Speicherplatz während der Kompilierung	1500 Mbyte (lokale Platte empfohlen)

Für 1.6/alpha:

Kompletter Satz aller Programme	1300Mbyte (NFS ok)

Die Angabe *NFS* steht für »Network File System« und bedeutet, daß dieser Teil des Package-Systems auch über ein Netzwerk gemeinsam nutzbar ist.

Ein Package wird deinstalliert, sobald aus diesem ein Binär-Package erzeugt wird. Auch werden alle Arbeitsverzeichnisse mit den Quellen gelöscht. Das heißt, es wird kein erheblicher Plattenplatz verbraucht.

5.3 NetBSD-Pakete und KDE

Wer einen NetBSD-Rechner als Desktop-System einsetzt und möglichst nicht die Kommandozeilen-Tools nehemen will, kann relativ komfortabel die Paketverwaltung auch mit dem KDE-Programm *kpackage* durchführen. Um *kpackage* zu verwenden, muß nicht das gesamte KDE-System installiert sein, es befindet sich im Paket »kdeadmin«, das »arts«, »kdebase« und »kdelibs« (und natürlich QT 3 und diverse zusätzliche Bibliotheksabhängigkeiten) benötigt.

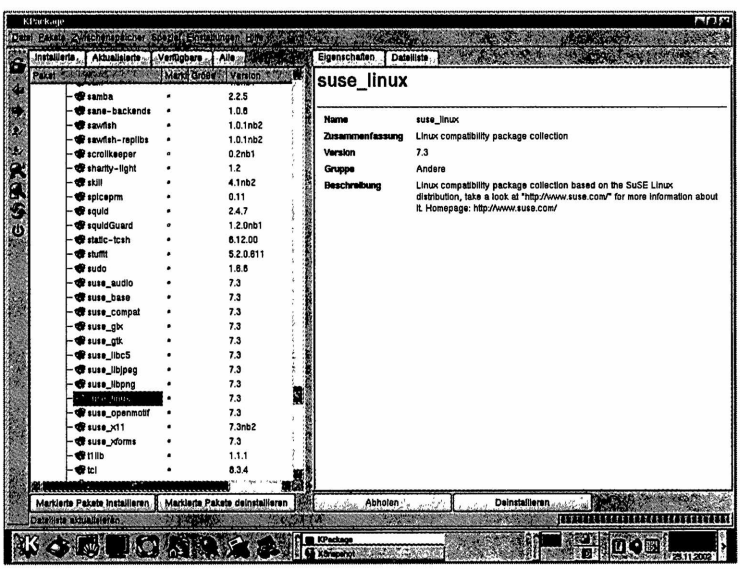


Abb. 5.2: Der BSD-Pakettree in »/var/db/pkg« wird perfekt verarbeitet

kpackage ist ein Multitalent, das mit BSD-Paketen, Debian-APT-Installationen und einigem mehr umgehen kann. Welche Paketformate verwendet werden sollen, wird über den Optionen-Dialog (Abb. 5.3) eingestellt. Die Angabe »BSD« in den Optionen von *kpackage* ist aber mit Vorsicht zu genießen. Man sieht schon an der Voreinstellung des Speicherorts für »Ports«, daß das Ports-System von FreeBSD als Vorbild genommen wurde. Wählt man (Abb. 5.4) das Package-Source-Verzeichnis (»/usr/pkgsrc«) aus, ist es aus!

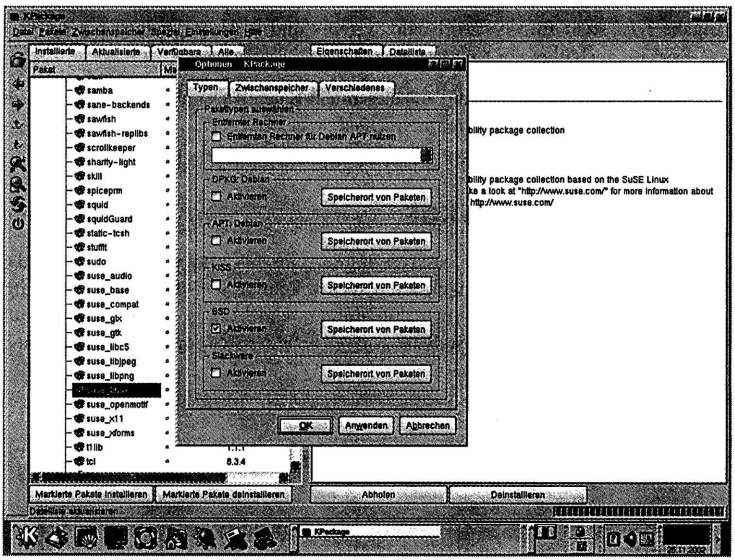


Abb. 5.3: kpackage muß für BSD konfiguriert werden

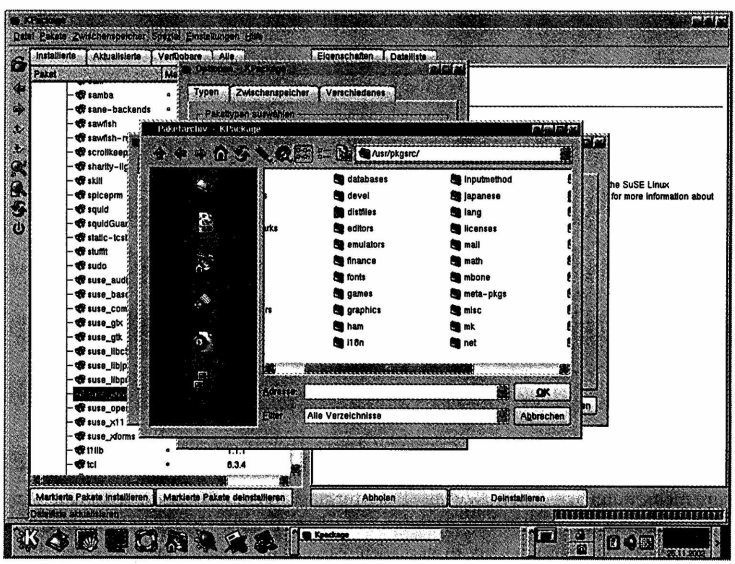


Abb. 5.4: Die Auswahl des Paketverzeichnisbaums

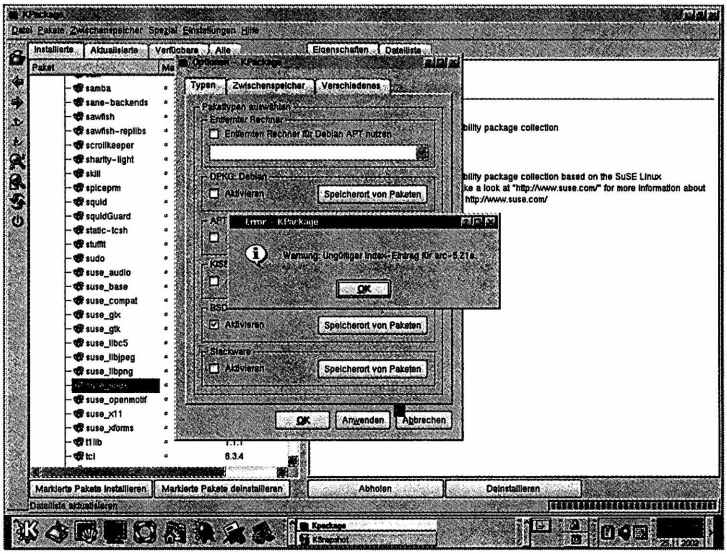


Abb. 5.5: Diese Fehlermeldung sieht man für jede Zeile der Datei INDEX

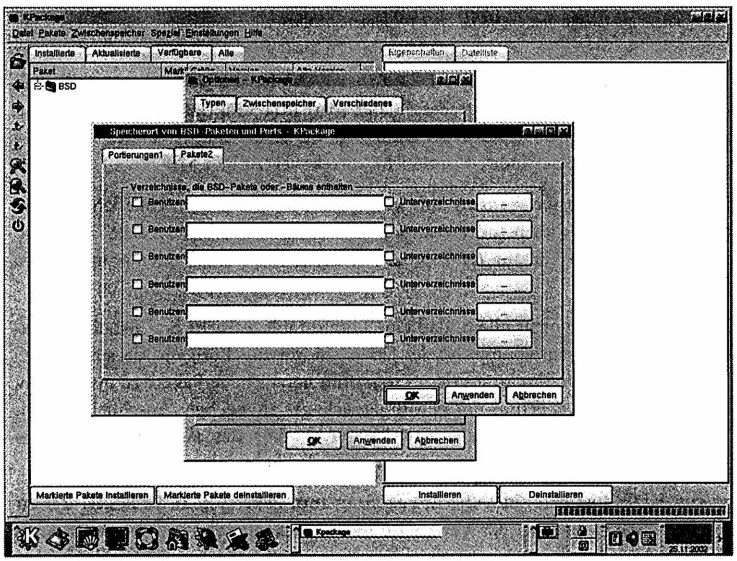


Abb. 5.6: Die Auswahl des Suchpfads für Binärpakete funktioniert, nur »Portierungen« sollte man nicht verwenden

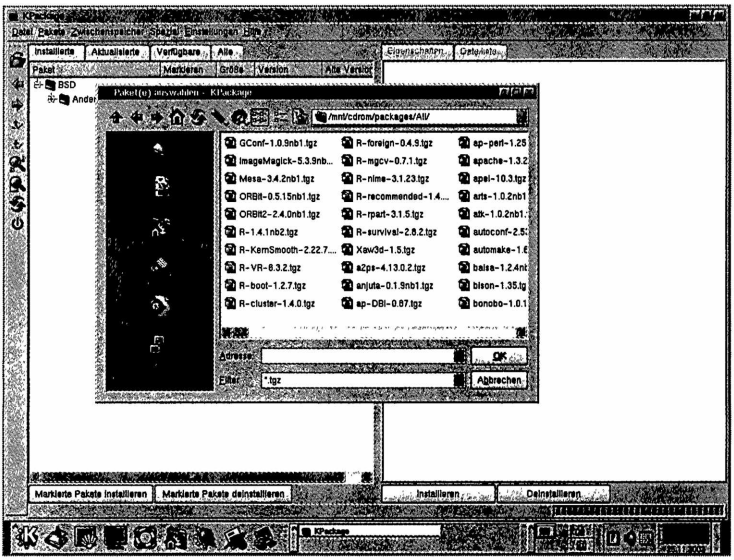


Abb. 5.7: Auswahl zu installierender Binärpackages. kpackage von KDE 3.0.3 ist da noch nicht auf dem neuesten Stand, da nur ».tgz«-Dateien, aber keine ».tbz«-Dateien zur Auswahl stehen. Das kann nicht nur bei NetBSD zu Problemen führen

kpackage versucht nun die Datei »INDEX« zu analysieren. Das geht zwar – falls man diese Datei mit *make index* angelegt hat – aber so richtig funktionieren will es nicht. Man darf nach dem Schließen des Auswahldialogs für jede Zeile der Datei »INDEX« bestätigen, daß die Zeile der Index-Datei nicht korrekt sei (Abb. 5.5).

Es ist in diesem Fall besser, *kpackage* mit *xkill* abzuschießen und anschließend neu zu starten. Man sollte also diesen Installationsdialog auf keinen Fall mit NetBSD verwenden. Das gilt auf jeden Fall für KDE 3.0.3. Vielleicht wird der Fehler ja in neueren Versionen behoben.

Praktisch an *kpackage* ist nicht allein, daß eine komfortable Ansicht der installieren Pakete inklusive deren Beschreibung und der Übersicht der zugehörigen Dateien geboten wird. Es ist auch möglich, direkt aus dem Programm heraus neue Pakete zu installieren. Dazu wird zuerst ein Verzeichnis mit den Binärpackages angegeben. Anschließend läßt sich aus diesem Verzeichnis ein Paket auswählen (Abb. 5.7), das dann analysiert, angezeigt und bei Bedarf auch getestet und installiert werden kann.

kpackage von KDE 3 ist eine sinnvolle Erweiterung zum Paketsystem. Es besitzt derzeit einige kleinere Schwächen bei der Verarbeitung des Package-Source-Verzeichnisses bei NetBSD. Bei der schnellen Weiterentwicklung von KDE sind dies Fehler aber sicher bald behoben. Dadurch, daß es sehr einfach zu bedienen ist und bei der eigentlichen Installation im Hintergrund die Paketverwaltungstools des Betriebssystems eingesetzt werden, treten auch keine Kompatibilitätsprobleme auf.

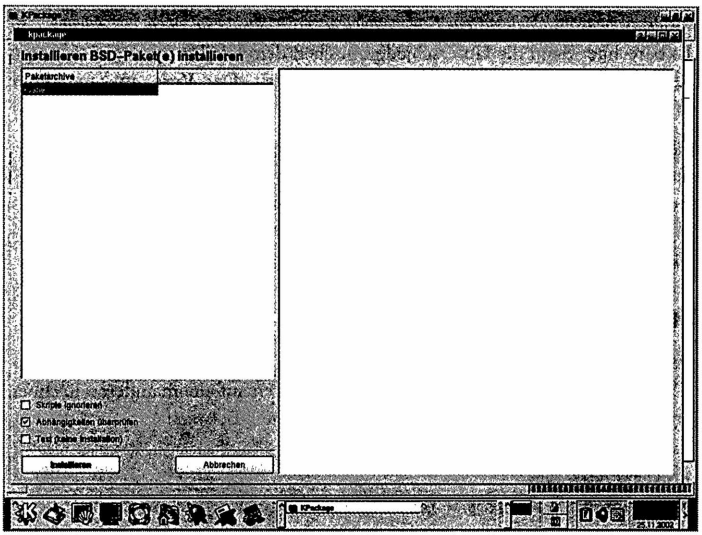


Abb. 5.8: Das Binärpackage von xsane wurde zur Installation ausgewählt

5.4 Aktualisieren des Package Source

5.4.1 cvs

Um zu beliebiger oder automatisierter Zeit die Hierarchie zu aktualisieren, bietet sich *cvs* – Control Versioning System – an. Es ändert Dateien, fügt neue Verzeichnisse oder Dateien hinzu, löscht aber auch Dateien. Führen Sie dazu folgende Befehle (als Benutzer *root*) aus:

Initiales Laden des Trees, alternativ zum Laden per FTP:

```
%> setenv CVSROOT anoncvs@anoncvs.netbsd.org:/cvsroot
%> setenv CVS_RSH ssh
%> cd /usr
%> cvs checkout -P pkgsrc
```

Update:

```
%> setenv CVSROOT anoncvs@anoncvs.netbsd.org:/cvsroot
%> setenv CVS_RSH ssh
%> cd /usr/pkgsrc
%> cvs -q -z3 update -P -d
```

Dies verbindet sich anhand der Daten in »/usr/pkgsrc/CVS« mit dem cvs-Server von NetBSD. Es berechnet dann die Unterschiede zwischen dem lokalen und dem entferten System, lädt und installiert neue und veränderte Dateien. Der Vorgang kann unter Umständen recht lange dauern.

Der diskutierbare Nachteil dieser Form der Package-Installation ist, daß man die originale Software im Sourcecode laden muß. Der unbestreitbare Vorteil ist, daß dieser Vorgang architekturunabhängig ist. Ein *make* beziehungsweise *make install* lädt diese Archive aus dem Internet. Ein explizites *make fetch* – egal, in welchem der Packages oder Pakkage-Hauptverzeichnisse (etc »/usr/pkgsrc/www«) – lädt die Distfiles explizit.

5.4.2 Offline aktualisieren

Wenn Sie keine permanente Anbindung an das Internet haben, genügt ein *make fetch-list*, um ein Shellskript zu erzeugen, mit dem Sie die Sourcen auf einem anderen Rechner (etwa in der Firma, mit hoher Bandbreite) nachladen können. Beispiel:

```
%> cd /usr/pkgsrc/www/lynx
%> make fetch-list DISTFILE=/tmp > fetch-lynx.sh
```

Kopieren oder mailen Sie die Datei »fetch-lynx.sh« auf einen Rechner mit hoher Bandbreite und führen Sie sie aus: *sh fetch-lynx.sh*. Die Anweisung lädt den Quelltext und die benötigten Patches und verwendet dabei mehrere Server, falls die Hauptseite nicht verfügbar sein sollte. Die Dateien werden unter »/tmp« abgelegt – sorgen Sie auf dem anderen Rechner dafür, daß ein solches Verzeichnis angelegt und für Sie beschreibbar ist. Gegebenenfalls passen Sie den Parameter für *DISTFILES* an.

Alternativ können Sie auch das komplette Source-Verzeichnis laden. Zuhause:

```
% cd /usr/pkgsrc
% make fetch-list FETCH_CMD=wget DISTDIR=/tmp/distfiles >/tmp/fetch.sh
```

Kopieren Sie beispielsweise per *scp* an den anderen Ort mit der hohen Bandbreite:

```
% scp /tmp/fetch.sh work:/tmp
```

In der Firma oder einem Ort mit hoher Bandbreite führen Sie das Skript aus:

```
% sh /tmp/fetch.sh
```

Das Verzeichnis »/tmp/distfiles« enthält nun die Distfiles. Diese nehmen Sie wieder mit nach Hause und legen sie in »/usr/pkgsrc/distfiles« ab. Rechnen Sie damit, daß die Distfiles zusammen ein paar 100 Mbyte groß sind.

Sind sie in der glücklichen Lage, einen NetBSD-Rechner mit guter Internet-Anbindung zu besitzen und möchten tatsächlich alle Distfiles laden, auch die, die nicht für die Ar-

chitektur ihres Rechners bestimmt sind, können Sie dies mit der obigen Variante *make fetch-list* tun oder mit diesem Befehl:

```
%> make mirror-distfiles
```

Wenn Sie die Lizenzbestimmungen beachten und die Sourcen trotzdem alle laden möchten, geschieht dies mit:

```
%> make fetch NO_IGNORE=yes
```

Damit ignorieren Sie, zumindest beim Laden der Software, deren Lizenz.

5.5 Überblick verschaffen

Einen erheblichen Vorteil hat das Installieren der Package-Sourcen noch: man kann ein lokales Info-Zentrum aufbauen und die Sammlung bequem durchforsten.

Führen Sie ein *make readme* im Package-Verzeichnis »/usr/pkgsrc« durch. Dies erzeugt eine Datei namens »README.html« im Hauptverzeichnis und allen Package-Verzeichnissen. Damit ist ein Überblick über verfügbare Packages geschaffen nebst einem (einzeiligen englischsprachigen) Kommentar über die Funktion des Packages sowie einem Verweis auf eine umfangreichere Information zum Package.

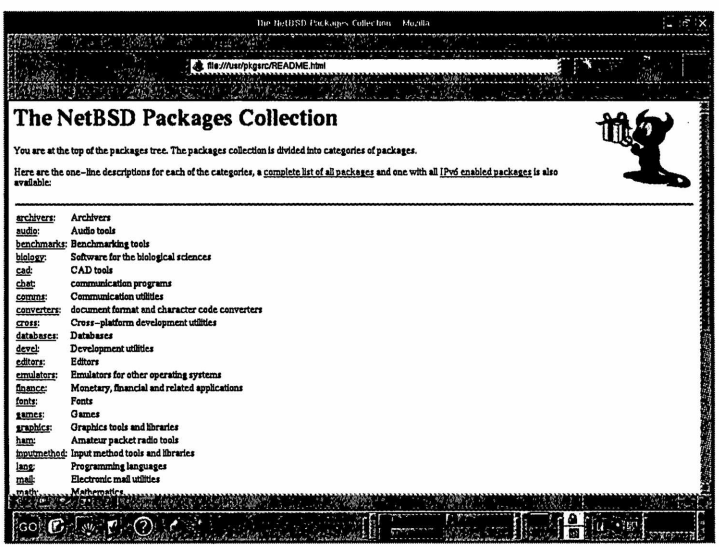


Abb. 5.9: Die HTML-Readme-Datei im Paket-Hauptverzeichnis. Hier im Mozilla-Browser und KDE 3 (die XP-Dekorationen sind nicht im Standard-Umfang von KDE 3 enthalten)

Die damit erzeugten Dateien können mit einem Web-Browser wie *lynx* (»/usr/pkg src/www/lynx«) oder Mozilla (»/usr/pkgsrc/www/mozilla«) angesehen werden. Verwenden Sie dazu die Option *lokale Dateien öffnen*.

Stets aktuell gehalten ist diese Information auch unter der bereits oben angebenen URL *ftp://ftp.netbsd.org/pub/NetBSD/packages/pkgsrc/README.html* abrufbar.

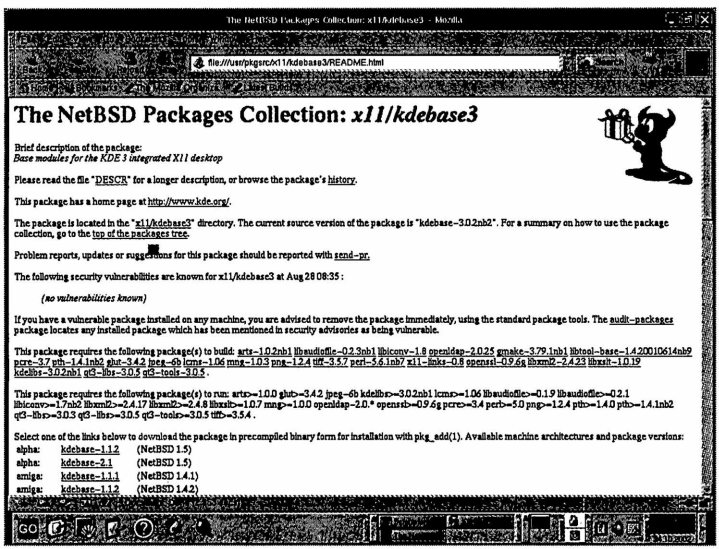


Abb. 5.10: Die Angaben in der README.html zu einem Package sind ausführlich und informativ

Eine weitere Möglichkeit ist die Erzeugung eines lokalen Index aller Packages mit dem Befehl *make index*. Dieser Befehl erzeugt einen Index »/usr/pkgsrc/INDEX«, der mit *make print-index | more* angezeigt werden kann. Achtung, die Indizierung dauert sehr lange, auf einem AMD K6/2 400 mehrere Tage! Aber diese Zeitdauer ist eigentlich logisch: Mehrere tausend Verzeichnisse müssen rekursiv auf Abhängigkeiten geprüft werden, um für jedes Paket eine komplette Dateiliste aufbauen zu können. So etwas muß sehr lange dauern. Drücken Sie auch während des Dateiaufbaus nicht [Ctrl][C], wenn Sie vielleicht der Meinung sind, daß die vorhandenen Paketzeilen ausreichen. Das Make-Skript fängt die Tastenkombination ab und reinigt vor dem Beenden das Verzeichnis. Zu diesen Reinigungsarbeiten gehört auch das Löschen der unvollständig angelegten Datei »INDEX«.

Mit dem Befehl *make search key=<Schlüsselwort>* kann die Hierarchie ebenfalls durchsucht werden. Allerdings muß dafür zuvor die Datei »INDEX« mit *make index* angelegt worden sein. Fehlt die Datei, wird sie beim ersten Aufruf von *make search* automatisch erzeugt:

```
[/usr/pkgsrc] #make search key=kmap
Generating INDEX - please wait ..
```

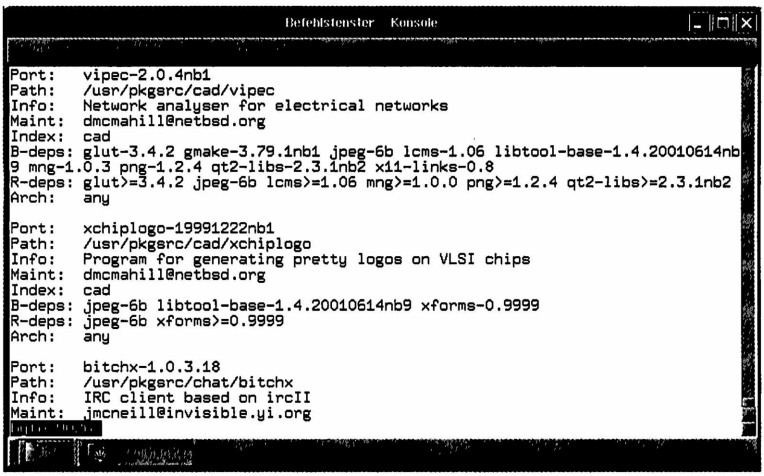


Abb. 5.11: Die Ausgabe von make search ist gut aufbereitet

Eine weitere Alternative für die Suche ist *pkglocate*. Dieses Skript befindet sich in »/usr/pkgsrc« und muß dort mit der Pfadangabe »./« aufgerufen werden. *pkglocate* durchsucht die »DESCR«-Beschreibungsdateien in den Verzeichnissen und liefert nur dann Resultate, wenn die Suchangabe in den entsprechenden Dateien enthalten ist. Die folgende Suche liefert deshalb ein recht mageres Resultat:

```
[/usr/pkgsrc]#./pkglocate kde
audio/kdemultimedia3/DESCR:The kdemultimedia module of KDE provides various
sound and movie
audio/krio/DESCR:This package is a kde-based front-end utility to the rio
devel/newmkdep/DESCR:This is a replacement for *BSD's "mkdep" command. Because
it is a C program
graphics/kdegraphics3/DESCR:The kdegraphics module of KDE is a collection of
graphics based programs
i18n/kde303-i18n/DESCR:WWW: http://i18n.kde.org/
x11/kdelibdocs2/DESCR:kab, kdecore, kdeui, kdeutils, kdoc-reference, kfile,
kfmlib, khtmlw,
x11/kdelibdocs3/DESCR:kab, kdecore, kdeui, kdeutils, kdoc-reference, kfile,
kfmlib, khtmlw,
x11/xpmicons/DESCR:A huge collection of icons, including Anthony's Icon
collection, kde,
```

Weiterführende Informationen

Die wichtigste Quelle für weitergehende Informationen ist »/usr/pkgsrc/Packages.txt«. Dort wird neben den hier gezeigten Vorgängen vor allem auch beschrieben, wie man ein sogenanntes *make bulk* erzeugt – einen vollständigen Satz aller vorkompilierten Packages – und diese auf CD-ROMs verteilt. Da dieser Vorgang sehr komplex und vor allem zeitraubend ist, wird er selten angewandt, ist aber eine ausgezeichnete Idee für Systemadministratoren mit höheren Anforderungen. Auch wird dort sehr ausführlich beschrieben, wie man eigene Packages erzeugt, was nahezu beliebige Software zu einem NetBSD-Package macht.

5.5.1 Packages und Firewall

Wenn Sie hinter einer Firewall sitzen und die Packages beziehungsweise Distfiles nicht direkt laden können, geben Sie einen Proxy-Server an (falls dieser zum Einsatz kommen muß), indem sie die entsprechenden Variablen setzen:

```
%> export ftp_proxy=ftp://orpheus.amdahl.com:80/
%> export http_proxy=http://orpheus.amdahl.com:80/
```

Das Package-System verwendet das bei NetBSD in der Basis-Installation installierte ftp-Programm, das sowohl die Protokolle FTP als auch HTTP beherrscht. Falls Sie passives FTP verwenden müssen, konfigurieren Sie das in der »/etc/mk.conf« mit der Option *PASSIVE_FETCH=1*.

5.5.2 Mailingliste

Es gibt eine eigenständige Mailingliste für Fragen rund um das Package-System. Tragen Sie sich mit diesem Kommando in die Liste ein:

```
%> echo subscribe tech-pkg | mail majordomo@netbsd.org
```

Die Mailing-Adresse der Liste selber lautet *tech-pkg@netbsd.org*

Bitte stellen Sie dort keine Fragen über die installierte Software, sondern nur über Probleme, Anregungen und Lösungen zum Package-System an sich. Hilfe zur installierten Software erhalten Sie dort selbst.

5.6 Lizenzbedingungen

Alle Packages beziehungsweise die mit einem Package installierbare Software werden durch eine jeweils eigene Lizenz in der Verbreitung und Nutzung der Software restriktiert. Das NetBSD-Package-System kennt fünf Formen der Lizenzpolitik, die sich auf das Package-System selbst beziehen und Auswirkungen auf die Installationsfähigkeit, Wiederverteilbarkeit und Ladbarkeit haben. Die Lizenz einer Software wird bei Ver-

wendung des Package-Sourcetree im Makefile in Variablen festgehalten. Folgende Werte können dort hinterlegt sein:

RESTRICTED

Diese Variable ist immer dann gesetzt, wenn es eine Beschränkung irgendeiner Art gibt. Sie enthält in der Regel den Grund der Beschränkung im (englischen) Klartext.

NO_BIN_ON_CDROM

Das Binär-Package dieser Software darf nicht auf einer CD-ROM verteilt werden. Das erlaubt es nicht, daß es eine sofort installierbare und lauffähige Version der Software auf CD-ROM geben kann. Ladbar vom Internet kann so ein Binär-Package allerdings sein. Die Variable wird auf *${RESTRICTED}* gesetzt.

NO_BIN_ON_FTP

Das Binär-Package dieser Software darf nicht auf einem FTP-Server im Internet verteilt werden. Diese Angabe erlaubt es nicht, daß es eine sofort installierbare und lauffähige Version der Software im Internet verfügbar gehalten werden darf. Die Variable wird auf *${RESTRICTED}* gesetzt.

NO_SRC_ON_CDROM

Das Distfile dieser Software darf nicht auf CD-ROM verteilt werden. Das erlaubt es nicht, daß der Sourcecode der Software auf CD-ROM verfügbar gehalten werden darf. Die Variable wird auf *${RESTRICTED}* gesetzt.

NO_SRC_ON_FTP

Das Distfile dieser Software darf nicht auf einem FTP-Server im Internet verteilt werden. Es ist also nicht erlaubt, den Quelltext der Software auf Mirror-Servern im Internet zu verteilen. Oft ist hier dann nur das Laden von der originalen Verteilstelle der Software möglich. Die Variable wird auf *${RESTRICTED}* gesetzt.

Die beschriebenen Einschränkungen bedeutet unter anderem, daß das Package-System von NetBSD einige Packages nicht als Binär-Packages im Internet oder auf CD-ROM zur sofortigen Installation anbieten darf.

Ist die Lizenz durch die Variable *RESTRICTED* reglementiert, müssen Sie bei Verwendung des Package-Sourcetree vor dem Start von *make install* oder *make package* gegebenenfalls die Lizenz explizit akzeptieren, indem Sie die Variable *ACCEPTABLE_LICENSES* in der Konfigurationsdatei »/etc/mk.conf« erweitern:

```
ACCEPTABLE_LICENSES+=jdk13-license
ACCEPTABLE_LICENSES+=shareware
```

Der erste Eintrag ist beispielsweise für die Installation des Java-1.3-Packages relevant (eine sehr restriktive Lizenz von SUN Microsystems). Der zweite Eintrag regelt zum Beispiel die Installation des Packages *xv*, das Shareware ist.

Wenn Sie hier einen Eintrag hinzufügen, sind Sie verantwortlich in eigener Person für die Einhaltung der Lizenzbestimmungen der Dritt-Software, die in einem Package verteilt und zur Verfügung gestellt wird.

Sie finden den Text aller das Paketsystem betreffenden Lizenzen im Verzeichnis »/usr/pkgsrc/licenses«.

5.7 Fehlermeldungen

Folgende Fehlermeldungen können auftreten, wenn eine Installation aus dem Package-Sourcetree gestartet wird:

```
===> <NAME> has an unacceptable license: shareware.
===>     To build this package, add this line to your /etc/mk.conf:
===>     ACCEPTABLE_LICENSES+=shareware
```

Die von Ihnen gewünschte Installation der Software des Package *<NAME>* enthält eine von Ihnen noch nicht autorisierte Lizenzpolitik. Passen Sie dazu die Datei »/etc/mk.conf« an. Danach kann mit *make install* die Software installiert werden.

```
*** If you really wish to overwrite the old package of <NAME>
*** without deleting it first, set the variable "FORCE_PKG_REGISTER"
*** in your environment or the "make install" command line.
*** Error code 1
```

Es ist in diesem Fall normalerweise angeraten, die Software erst zu deinstallieren, mit *make delete* oder *pkg_delete <NAME>* und dann mit *make install* neu zu installieren.

Andere Fehlermeldungen sind leider unergiebiger:

```
<Lange Liste von Fehlern im C-Sourcecode>
*** Error code 1
```

In diesem Fall hilft vielleicht ein cvs-Update des Package-Sourcetrees, da das Package-System vielleicht inzwischen einen Patch bereitstellt. Oder Sie löschen das Distfile und starten die Installation neu. Unter Umständen ist nämlich eine neue Version der Software verfügbar oder das Distfile defekt. Hilft das alles nichts, haben Sie vielleicht einen Bug entdeckt. Melden Sie ihn über die Mailingliste oder am besten mit Hilfe des offiziellen Bugreport-Tools *send-pr*, das jedem NetBSD als Programm beiliegt. Wählen Sie die Kategorie *pkg* und versuchen Sie, die genaue Fehlermeldung zu notieren.

5.8 Das Bauen eigener Package-Source-Verzeichnisse

von Jörg Braun

Um eigene Pakete zu bauen, benötigt man naturgemäß mehr Informationen als für deren reine Verwendung. Außerdem gibt es verschiedene Wege, um ein neues NetBSD-Paket zu schnüren. Dieses Unterkapitel zeigt mehrere unterschiedliche Wege auf, wie man ein neues Package-Source-Verzeichnis baut und dann daraus auch ein eigenes Binär-Package erzeugt.

Grundsätzlich kann man (fast) jede Software mit dem normalen GNU-Konfigurations- und -Installationslauf ins System einbinden. Dazu besitzen die meisten Programme mehrere vorbereitete Werkzeuge. An erster Stelle ist die Datei »configure« zu nennen. Sie dient – wie der Name schon sagt – der Konfiguration einer Software. *GNU configure* paßt ein Softwarepaket, das in Form von Quelldateien vorliegt, an ein Betriebssystem an. Dazu wird die Make-Datei »Makefile« angelegt, die anschließend bei einem Aufruf von *make* ausgewertet wird.

Eine »normale« Installation ohne das NetBSD-Paketsystem läuft also in den Schritten

```
tar -xzf <Archiv_der_Software>.tgz
cd <Archiv_der_Software>
./configure
make
make install
```

ab. Wenigstens der letzte Schritt muß dabei mit Administratorrechten durchgeführt werden.

Diese Schritte sind im *make* des Package-Source-Systems von NetBSD wiederzufinden:

```
make extract
make configure
make
make install
```

Das *make* des Package-Source-Systems enthält hierbei einige zusätzliche Schritte wie *make fetch*, *make patch* oder auch *make package*.

Grundsätzlich gilt es also bei den allermeisten Paketen nur dafür zu sorgen, die Schritte des *GNU configure* und *make* mit denen von NetBSD zu korrelieren. Die Korrelation wird dabei in der Datei *Makefile* oder auch in zusätzlichen Dateien vorgenommen. Die Steuerung erfolgt über Umgebungsvariablen, die anschließend ausgewertet werden. Hier eine typische Makedatei eines relativ einfachen Pakets:

```
# $NetBSD$

DISTNAME=       latex2rtf-1.9.13
```

```
CATEGORIES=     print
MASTER_SITES=   ${MASTER_SITE_TEX_CTAN}

MAINTAINER=     jb@toolbox-mag.de
COMMENT=        Translator from LaTeX to RTF

IGNOREFILES=    ${DISTNAME}${EXTRACT_SUFX}

.include "../../mk/bsd.pkg.mk"
```

Sie finden dieses Paket nicht im NetBSD-Package-Sourcetree, da es privat von dem entsprechenden FreeBSD-Port konvertiert wurde. Für diese Konvertierung enthält NetBSD das Package-Source-Verzeichnis *port2pkg* von Hubert Feyrer, das die Grunddateien eines FreeBSD-Ports nach NetBSD überträgt und dabei die entscheidenden Bereiche umbenennt. Dabei werden auch in der Make-Datei selbst einige Werte umbenannt.

FreeBSD-Datei	NetBSD-Datei	Funktion
pkg-descr	DESCR	Beschreibungsdatei
pkg-plist	PLIST	Dateiliste
pkg-comment	Kommentarfeld in der Datei Makefile	Paketkurzkommentar
./files	./files	Zusatzdateien
./patches	./patches	BSD-Patches
distinfo	distinfo	Checksummendatei

Tabelle 5.4: Steuerdateien bei NetBSD und FreeBSD

```
# New ports collection makefile for:    latex2rtf
# Date created:         14 Dec 1999
# Whom:                 Joerg Wunsch <joerg@FreeBSD.org>
#
# $FreeBSD: ports/print/latex2rtf/Makefile,v 1.5 2002/08/28 joerg Exp

PORTNAME=       latex2rtf
PORTVERSION=    1.9.13
CATEGORIES=     print
MASTER_SITES=   ${MASTER_SITE_SOURCEFORGE} \
                ${MASTER_SITE_TEX_CTAN}
MASTER_SITE_SUBDIR=     ${PORTNAME} support/latex2rtf
MAINTAINER=     joerg@FreeBSD.org
MAN1=           latex2rtf.1

.include <bsd.port.mk>
```

Die beiden Programmversionen der Make-Datei unterscheiden sich zwar, sie zeigen aber einige Übereinstimmungen. Einige Grundbezeichner lauten anders, sie werden aber von *port2pkg* automatisch korrigiert. Wichtig ist, daß, während FreeBSD für den einzeiligen Kommentar eine eigene Datei verwendet, dieser in der NetBSD-Make-Datei integriert ist.

Um einen wie in diesem Fall sehr einfachen Port zu übernehmen, reicht der Download des FreeBSD-Ports in ein eigenes Verzeichnis und der Aufruf des Programms *port2pkg*:

```
port2pkg <Quellverzeichnis> <Zielverzeichnis>
```

Leider ist das aber nicht der Normalzustand. Die beiden Verzeichnisse unterscheiden nämlich bei einigen wichtigen Variablen. Die beiden folgenden Listings zeigen zuerst den FreeBSD-Port für die deutschen i18n-Erweiterungen von KDE 3.0.3

```
# New ports collection makefile for:    kde3-i18n-de
# Date created:                         05 April 2002
# Whom:                                 Lauri Watts <lauri@kde.org>
#
# $FreeBSD: ports/german/kde3-i18n/Makefile,v 1.9 2002/08/25 will Exp
#

PORTNAME=       kde-i18n
PORTVERSION=    ${KDE_VERSION}
CATEGORIES?=    misc kde
MASTER_SITES=   ${MASTER_SITE_KDE}
MASTER_SITE_SUBDIR=     stable/${PORTVERSION}/src/kde-i18n
DIST_SUBDIR=    KDE

MAINTAINER?=    kde@FreeBSD.org

BUILD_DEPENDS=  xgettext:${PORTSDIR}/devel/gettext

USE_KDELIBS_VER=3
USE_BZIP2=      yes
USE_GMAKE=      yes
GNU_CONFIGURE=  yes
KDE_I18N=       yes

.include "${.CURDIR}/../../x11/kde3/Makefile.kde"

.include <bsd.port.mk>
```

Und nun die Anpassung an NetBSD:

```
# $NetBSD: Makefile.inc,v1.0 2002/08/31 13:50:00 jb Exp $
# Include make file for kde-i18n packages
#
# This file misses only KDE version and language definitions
# These variables are defined in the main Makefile which includes
# this file
# LANG=
# KDEVER=

DISTNAME=       kde-i18n-${LANG}-${KDEVER}
CATEGORIES=     x11 kde

# New additional directory i18n/ in KDE 3.0.3:
.if ( ${KDEVER} == 3.0.2 )
MASTER_SITES=${MASTER_SITE_KDE:=stable/${KDEVER}/src/}
.else
MASTER_SITES=${MASTER_SITE_KDE:=stable/${KDEVER}/src/i18n/}
.endif

EXTRACT_SUFX=   .tar.bz2
WRKSRC=         ${WRKDIR}/kde-i18n-${LANG}

MAINTAINER=     jb@toolbox-mag.de
HOMEPAGE=       http://i18n.kde.org/
COMMENT=        Localized messages and documentation for KDE3 for ${LANG}

.include "../../x11/kde3/Makefile.kde3"

DEPENDS=        gettext>=0.10.35:../../devel/gettext
# This is the correct dependency:
#DEPENDS+=      kdebase-${KDEVER}:../../x11/kdebase3
# This was tested with several KDE 3.0.3 packages on KDE 3.0.2 and
# it works:
DEPENDS+=       kdebase>=3.0.1:../../x11/kdebase3

USE_KDELIBS_VER=3
USE_BZIP2=      yes
USE_GMAKE=      yes
USE_X11BASE=    yes
GNU_CONFIGURE=  yes
CONFIGURE_TARGET=

CONFIGURE_ARGS += kde_datadir=/usr/X11R6/share/kde/apps
CONFIGURE_ARGS += kde_locale=/usr/X11R6/share/kde/locale
CONFIGURE_ARGS += kde_htmldir=/usr/X11R6/share/doc/kde/HTML

.include "../../mk/bsd.pkg.mk"
```

Was als erster Unterschied ins Auge fällt, ist, daß die Quellarchive im Format ».tar.bz2« vorliegen. Bei NetBSD ist dafür die Information *EXTRACT_SUFX,* bei FreeBSD die Variable *USE_BZIP2* zuständig. Außerdem verweisen diese Pakete noch auf andere KDE-Verzeichnisse, deren Make-Dateien ebenfalls berücksichtigt werden müssen.

Hier hilft also ein einfaches *port2pkg* nicht mehr. Außerdem befinden sich am Schluß der Datei mehrere Zeilen mit dem Eintrag *CONFIGURE_ARGS* +=. Ohne sie funktionieren bei NetBSD keine KDE-Pakete. Der Grund ist, daß NetBSD bei der Installation der Dateien sich nicht an die Voreinstellungen von KDE hält. Normalerweise werden (und so auch bei FreeBSD) die Dateien direkt in Unterverzeichnisse von »/usr/local« installiert. Die Spracherweiterungen kommen folgerichtig direkt nach »/usr/local/share/locale« und die (lokalisierte) Hilfe nach »/usr/local/share/doc«. Bei NetBSD wandern diese Dateien nach /usr/X11R6/share/kde/locale und »/usr/X11R6/share/doc/kde/«. Der *configure*-Lauf muß dem Rechnung tragen.

Für die reine Installation der Dateien ohne die Möglichkeit des Bauens eines Pakets und die Deinstallation würde das reichen. Da sich die Pfade unterscheiden, muß aber auch die Paketliste (»pkg-plist« bei FreeBSD und »PLIST« bei NetBSD) angepaßt werden. Dazu muß die von FreeBSD stammende Datei in einen Editor geladen und durch Suchen&Ersetzen angepaßt werden. Das Original (hier sind nur die ersten Zeilen gezeigt)

```
share/apps/ktuberling/sounds/de/abzeichen.wav
share/apps/ktuberling/sounds/de/auge.wav
share/apps/ktuberling/sounds/de/augenbraue.wav
share/apps/ktuberling/sounds/de/brille.wav
share/apps/ktuberling/sounds/de/fliege.wav
share/apps/ktuberling/sounds/de/haare.wav
share/apps/ktuberling/sounds/de/hut.wav
share/apps/ktuberling/sounds/de/ohr.wav
share/apps/ktuberling/sounds/de/ohrring.wav
share/apps/ktuberling/sounds/de/schnurrbart.wav
...
```

muß zum Schluß so aussehen:

```
share/kde/apps/ktuberling/sounds/de/abzeichen.wav
share/kde/apps/ktuberling/sounds/de/auge.wav
share/kde/apps/ktuberling/sounds/de/augenbraue.wav
share/kde/apps/ktuberling/sounds/de/brille.wav
share/kde/apps/ktuberling/sounds/de/fliege.wav
share/kde/apps/ktuberling/sounds/de/haare.wav
share/kde/apps/ktuberling/sounds/de/hut.wav
share/kde/apps/ktuberling/sounds/de/ohr.wav
share/kde/apps/ktuberling/sounds/de/ohrring.wav
share/kde/apps/ktuberling/sounds/de/schnurrbart.wav
...
```

Das Grundverzeichnis (»/usr/local/« gegenüber »/usr/X11R6«) darf nicht angegeben sein. Es wird über die generische Makedatei festgelegt.

Sie sehen an diesem Beispiel, welche Schritte beim Umsetzen eines Pakets zu beachten sind. Die Sprachpakete von KDE sind dabei aber ein nicht besonders einfaches Beispiel.

Es gibt aber auch Pakete, die man selbst sehr einfach erzeugen kann, beispielsweise wenn man sicherstellen will, daß ein komplettes Programmpaket mit allen Abhängigkeiten und Zusatzprogrammen installiert wird. Solche Meta-Packages finden Sie in »/usr/pkgsrc/meta-pkgs« beispielsweise für den Window Manager WindowMaker und für die NetBSD-Dokumentation. Hier werden keine eigentlichen Pakete installiert, sondern nur Abhängigkeiten automatisch aufgelöst.

Ein Paket für den neuen Rox-Desktop würde beispielsweise alle Rox-Anwendungen installieren.

```
# $NetBSD: Makefile,v 1.2 2002/06/13 08:22:01 jb Exp $
#
DISTNAME=       rox-desktop-1.2.0
CATEGORIES=     meta-pkgs
MASTER_SITES=   # empty
DISTFILES=      # empty

MAINTAINER=     jb@toolbox-mag.de
HOMEPAGE=       ${MASTER_SITE_SOURCEFORGE:=rox/}
COMMENT=        provides a RISC OS like desktop

DEPENDS+=       rox-base>=1.0:../../sysutils/rox-base
DEPENDS+=       rox>=1.2.0:../../sysutils/rox
DEPENDS+=       rox-archive>=0.1.2:../../sysutils/rox-archive
DEPENDS+=       rox-edit>=0.1:../../sysutils/rox-edit
DEPENDS+=       rox-lib>=0.1:../../sysutils/rox-lib
DEPENDS+=       rox-memo>=1.0.0:../../sysutils/rox-memo
DEPENDS+=       rox-session>=0.1.14:../../sysutils/rox-session
DEPENDS+=       rox-system>=0.1.2:../../sysutils/rox-system
DEPENDS+=       rox-wallpapers>=0.1.1:../../sysutils/rox-wallpapers
DEPENDS+=       rox-wrappers>=1.0.2:../../sysutils/rox-wrappers-1.0.2

EXTRACT_ONLY=   # empty
NO_CHECKSUM=    yes
NO_PATCH=       yes
NO_CONFIGURE=   yes
NO_BUILD=       yes
do-install:     # empty

.include "../../mk/bsd.pkg.mk"
```

Die »PLIST«-Datei kann bei solchen Paketen entweder leer (also 0 Byte groß) sein oder nur den entsprechenden Kommentar enthalten:

```
@comment $NetBSD: PLIST,v 1.1 2002/06/13 08:22:01 rh Exp $
@comment this plist intentionally left empty
```

Der Grundaufbau einer entsprechenden Make-Datei ist – wie man an den vorangegangenen Beispielen sieht – wie folgt:

```
#$ NetBSD Makefile,v Versionsnummer Datum Uhrzeit Kürzel Exp$
DISTNAME=       Name des Pakets
PKGNAME=        Name des Packages, nur wenn abweichend von DISTNAME
PKGREVISION=    Zusätzliche Nummer der Revision des Pakets
CATEGORIES=     Software-Kategorien, mit Leerzeichen getrennt
MASTER_SITES=   Ursprungsort(e) der Quelldateien
DISTFILES=      Zusätzliche außer dem Hauptpaket benötigte Dateien
EXTRACT_SUFFX=  Dateiendung der Ursprungsdatei wenn nicht .tar.gz

MAINTAINER=     Name und E-Mail-Adresse des Paketbauers
HOMEPAGE=       Homepage des Quellpakets
COMMENT=        Einzeiliger Kommentar zum Paket

DIST_SUBDIR=    Eventuell abweichender Name des Arbeitsverzeichnisses
USE_X11BASE=    YES # wenn Ziel nicht nach /usr/pkg, sondern /usr/X11R6
DEPENDS+=       Für die Verwendung benötigte Packages
BUILD_DEPENDS+= Nur für den Buildvorgang benötigte Packages

# Fehlermeldungen und Abbruch bei nicht installierbaren oder defekten
# Paketen:
IGNORE=         "Dieses Paket kann nicht installiert werden"
BROKEN=         "Dieses Paket ist fehlerhaft"

# Besondere Maßregeln für die Weitergabe und Lizenzangaben:
RESTRICTED=             "Fehlermeldung die entsprechend angezeigt wird"
NO_SRC_ON_CDROM=        ${RESTRICTED}
NO_BIN_ON_CDROM=        ${RESTRICTED}
NO_BIN_ON_FTP=          ${RESTRICTED}
NO_SRC_ON_FTP=          ${RESTRICTED}
LICENSE=                Die Angabe abweichender Lizenzen

# Ausschluß von Hardware-Plattformen:
NOT_FOR_PLATFORM= Angabe von Plattformen die nicht unterstützt werden
ONLY_FOR_PLATFORM= Angabe ausschließlich unterstützter Plattformen

CONFIGURE_ARGS+= Angabe zusätzlicher Flags für configure
```

```
# Make-Labels für Zusatz- (pre-, post-) oder Ersatzarbeiten (do-)
# Die eigentlichen Arbeitszeilen müssen mit einem Tabulatorzeichen
# eingerückt sein:
pre-patch:
do-patch:
post-patch:
pre-configure:
do-configure:
post-configure:
pre-install:
do-install:
post-install:

# Die Hauptdatei muß immer zum Schluß eingebunden sein:
.include "../../mk/bsd.pkg.mk"
```

Dies sind nur die wichtigsten Steuervariablen für Pakete. Es gibt noch zahlreiche weitere, zum Beispiel für die Angabe, ob für den Konfigurationsvorgang *GNU configure* oder für den Make-Lauf *GNU Make* (*gmake*) oder *imake* des X11-Pakets zu verwenden ist.

5.9 Das Bauen eigener Binärpackages

von Jörg Braun

Wie baut man jetzt ein Package, wenn man keine Quellen zur Verfügung hat, beispielsweise wenn man ein Programm, das sich nicht mit dem *gcc* oder einem der anderen üblichen Compiler erzeugen läßt, weitergeben will? Das ist nicht ganz einfach, soll hier aber am Beispiel des Free-Pascal-Compilers erläutert werden.

Free Pascal ist ein für die verschiedenen Plattformen verfügbarer Pascal-Compiler mit an Borland Pascal und Delphi angelehnter Syntax. Der Compiler steht prinzipiell auch für NetBSD zur Verfügung. Nur kann er nicht unter NetBSD mit einem der herkömmlichen Entwicklungstools aus Sourcen erzeugt werden. Er muß dafür mit einer anderen Version von Free Pascal crosskompiliert werden. Dazu eignet sich am besten die Linux-Version, da sie in der Linux-Emulationsbox von NetBSD problemlos funktioniert. Auf das Crosscompile des Compilers soll hier nicht eingegangen werden. Nur soviel: Nach dem Übersetzen der diversen Programme (Compiler und Tools) werden diese normalerweise nach »/usr/local« installiert.

Da der Verzeichnisbaum in »/usr/local« unter NetBSD normalerweise leer ist, sind also alle Dateien zu Free Pascal gehörig. Ein *ls -R* in »/usr/local« listet alle Dateien in diesem Verzeichnisbaum auf und ist die Grundvoraussetzung für die Datei »PLIST«. In einem Editor muß die Ausgabe des Programms nur nachbearbeitet und dabei jeder Datei die Pfadangabe (»bin/« und »lib/«), vorangestellt werden:

```
@name fpc-1.0.6-pre
bin/bin2obj
bin/data2inc
```

```
bin/delp
bin/fd2pascal
bin/fpc
bin/fpcmake
bin/fprcp
bin/h2pas
bin/h2paspp
bin/plex
bin/postw32
bin/ppc386
bin/ppdep
bin/ppudump
bin/ppufiles
bin/ppumove
bin/ptop
bin/pyacc
bin/rstconv
bin/samplecfg
etc/fpc.cfg
lib/fpc/1.0.6/ppc386
lib/fpc/1.0.6/samplecfg
lib/fpc/1.0.6/msg/errord.msg
lib/fpc/1.0.6/msg/errore.msg
lib/fpc/1.0.6/msg/errores.msg
...
```

Nun muß ein neues Package-Source-Verzeichnis (beispielsweise »/usr/pkgsrc/devel/fpc«) angelegt werden. Hier werden eine Make-Datei, die Dateien »PLIST« und »DESCR« sowie ein Verzeichnis »work« benötigt.

In »work« werden die folgenden Dateien mit *touch* erzeugt:

```
.build_done
.build_info
.build_version
.buildlink_done
.configure_done
.extract_done
.install_done
.patch_done
```

Außerdem wird die Dateien »PLIST« und »DESCR« unter den Namen ».PLIST« und ».DESCR« ebenfalls in diesem Verzeichnis abgelegt.

Nun legt man ein Verzeichnis, korrespondierend mit dem Namen und der Versionsnummer in »work« an, bei Free Pascal 1.0.6 hieße es »fpc-1.0.6«.

In dieses Verzeichnis kopiert man den kompletten Verzeichnisbaum der Free-Pascal-Distribution, also Eins-zu-Eins in diesem Fall den Verzeichnisbaum von »/usr/local«.

Nun fehlen nur noch die zwei Dateien zur Größe der installierten Pakete. Die Gesamtgröße erfährt man durch einen Aufruf von *du* in »/usr/local«. Man trägt die Zahl in die Dateien ».SizeAll« und ».Size_Pkg« ein.

Damit ist man fast fertig. Benötigt wird nur noch eine angepaßte Makedatei. Sie sollte die folgenden Einträge besitzen:

```
# $NetBSD: Makefile,v 1.2 2002/06/13 08:22:01 jb Exp $
#

DISTNAME=       fpc-1.0.6
CATEGORIES=     devel
MASTER_SITES=   # empty
DISTFILES=      # empty

MAINTAINER=     jb@toolbox-mag.de
HOMEPAGE=       http://www.freepascal.org
COMMENT=        The NetBSD version of the Free Pascal compiler

NO_CHECKSUM=    yes

.include "../../mk/bsd.pkg.mk"
```

Da das Make-System kein Distfile angegeben hat, wird nicht versucht, die fehlende Datei herunterzuladen, außerdem ist im System vermerkt, daß alle anderen Läufe schon erfolgreich durchgeführt wurden. Damit wird jetzt – ohne daß ein Quellpaket existiert – das Paket geschnürt, wenn man *make package* aufruft.

Anschließend kann man das Softwarepaket NetBSD-gerecht distribuieren und der Anwender kann das Programm mitsamt der zugehörigen Dateien auch wieder aus dem System rückstandslos entfernen, was ja der Zweck der Übung war.

5.10 Der Grundaufbau des Package-Sourcetrees

von Jörg Braun

Bei NetBSD ist das Hauptverzeichnis für die Pakete normalerweise »/usr/pkgsrc«. Es ist zwar auch möglich, ein anderes Verzeichnis zu wählen, davon ist aber abzuraten, denn die verwendeten Make-Dateien in »/usr/pkgsrc/mk« verweisen auf Dateien in »/usr/share/mk«, die sonst eventuell nicht gefunden werden. Sollten Sie aus Platzgründen den Verzeichnisbaum an anderer Stelle anlegen müssen, empfiehlt es sich, einen symbolischen Link nach »/usr/pkgsrc« zu legen und anschließend von dort aus über den Link zu arbeiten.

Damit die Paketverzeichnisse funktionieren und relativ logisch aufgebaut sind, hat sich die Gliederung des Verzeichnisbaums in die folgenden Unterverzeichnisse bewährt:

archivers	Archivierungs- und Packprogramme
audio	Alles zu Soundein- und -ausgabe
benchmarks	Geschwindigkeits-Testprogramme
biology	Wissenschaftliche Programme nicht nur zu Biologie, sondern auch zu Chemie
cad	CAD-Software, die Auswahl ist hier sehr eingeschränkt
chat	Chat- und IRC-Software
comms	Kommunikationsprogramme und -protokolle
converters	Daten-Konvertierungsprogramme
corba	CORBA, bei NetBSD 1.6 ist dieses Verzeichnis leer
cross	Cross-Compiler und -Tools
databases	Datenbank-Server und -Clients
devel	Alles zur Software-Entwicklung mit Ausnahme der Programme, die sich in »lang« befinden
editors	Editoren, Textverarbeitungen
emulators	Betriebssystem-Emulatoren
finance	Finanzsoftware
fonts	Schriften und direkt schriftenbezogene Programme
games	Spiele
graphics	Grafikprogramme und -bibliotheken
ham	Software für Amateur-Radio
inputmethod	Eingabebezogene Software, vorwiegend für Fernost
japanese	Speziell an die japanische Schrift angepaßte Programme. Derzeit leer
lang	Programmiersprachen. Teilweise befinden sich solche Programme auch in »devel«
mail	Programme für E-Mails
math	Mathematik-Programme
mbone	Applikationen für Multi-Cast Backbone
meta-pkgs	Zusammenfassende Pseudopakete
misc	Diverses

news	News-Reader und -Server
parallel	Programme und Bibliotheken für Parallelverarbeitung
pkgtools	Besondere Pakettools
print	Alles zum Drucken
security	Sicherheitsrelevante Pakete, insbesondere Daemons und Clients
shells	Kommandozeilen-Shells
sysutils	Systemprogramme
textprocs	Textbearbeitungssoftware
time	Alles zum Thema Datum und Zeit
wm	WindowManager. Eine anderer Teil befindet sich in »x11«
www	Alles zum Thema WWW. Server, HTML-Browser und so weiter
x11	Alles zum Thema X11 das nicht in andere Verzeichnisse ausgelagert ist

Die folgenden Verzeichnisse dienen der Verwaltung:

CVS	Das übliche CVS-Verzeichnis befindet sich jeweils in allen Verzeichnissen des Baums
distfiles	Der Platz für die Quellpakete
licenses	Die die Programme im Paketverzeichnis betreffenden Lizenzen
mk	Das Grundverzeichnis mit den globalen Makedateien für das PkgSrc-Verzeichnis
packages	Das Verzeichnis mit den erzeugten Binärpaketen
templates	Dateien für die Erzeugung der Readmes

Es macht wenig Sinn, hier alle aktuellen Pakete aufzulisten, da der Verzeichnisbaum ständig wächst und Programme laufend aktualisiert werden. Es steht Ihnen auch frei, für eigene Paketverzeichnisse neue private Pfade aufzubauen. Damit dies gelingt, müssen Sie neue Verzeichnisse in die jeweils im übergeordneten Verzeichnis befindliche Datei »Makefile« aufnehmen. Die folgende Datei befindet sich in »/usr/pkgsrc«. Sie wurde privat um das Verzeichnis »i18n«, in dem die Spracherweiterungen für KDE abgelegt wurden, ergänzt:

```
# $NetBSD: Makefile,v 1.47 2002/05/31 15:26:50 seb Exp $
#
```

```
.include "mk/bsd.prefs.mk"


.ifdef SPECIFIC_PKGS
SUBDIR+=        ${SITE_SPECIFIC_PKGS}
SUBDIR+=        ${HOST_SPECIFIC_PKGS}
SUBDIR+=        ${GROUP_SPECIFIC_PKGS}
SUBDIR+=        ${USER_SPECIFIC_PKGS}
.else
SUBDIR += archivers
SUBDIR += i18n
SUBDIR += audio
SUBDIR += benchmarks
SUBDIR += biology
SUBDIR += cad
SUBDIR += chat
SUBDIR += comms
SUBDIR += converters
SUBDIR += cross
...
```

Eine Alternative zum direkten Eintrag von »i18n« wäre die Belegung beispielsweise der Variable *SITE_SPECIFIC_PKGS* mit dem Wert *i18n* gewesen. Diese Belegung erfolgt alternativ in der Datei »/etc/mk.conf« oder im Environment.

Generell gilt, daß das PkgSrc-Verzeichnis für alle Plattformen zur Verfügung steht, dabei sind plattformspezische Besonderheiten automatisch berücksichtigt. Wird beispielsweise auf einem Macintosh mit 680x0-Prozessor versucht, ein Programm zu übersetzen, das nicht für diese, sondern nur für andere Ports zur Verfügung steht, erscheint die folgende Meldung:

```
[/usr/pkgsrc/www/mozilla]#make
===> mozilla-1.1nb1 is not available for NetBSD-1.6-m68k
```

Diese oder eine adäquate Meldung sieht man auf etwas exotischeren Plattformen häufiger. Grund bei Mozilla ist die folgende Zeile in der Make-Datei:

```
# NetBSD-*-m68k builds, but "regchrome" dumps core.
NOT_FOR_PLATFORM=       NetBSD-1.4.*-* NetBSD-*-m68k
```

Sie bedeutet, daß Mozilla auf älteren NetBSD-Versionen und auf dem 68000-er nicht verwendet werden kann. Sie können solche Hinweise natürlich aus der Datei entfernen oder mit einem #-Zeichen auskommentieren, tun das aber auf eigene Gefahr. Es gibt auch die andere Variante:

```
ONLY_FOR_PLATFORM=*-*-i386 *-1.4[L-Z]-sparc *-1.[5-9]*-sparc \
                  *-[2-9]*-sparc
```

Diese Zeile (im Original nicht getrennt) stammt aus der Makedatei des Adobe Acrobat Readers und führt zur Meldung:

```
[/usr/pkgsrc/print/acroread] #make
===> acroread-4.05 has an unacceptable license: adobe-acrobat-license.
===>     To build this package, add this line to your /etc/mk.conf:
===>     ACCEPTABLE_LICENSES+=adobe-acrobat-license
===> acroread-4.05 is not available for NetBSD-1.6-m68k
```

Sie sollten also beim Versuch des Einbinden eines Pakets genau schauen, ob es wirklich zur Verfügung steht.

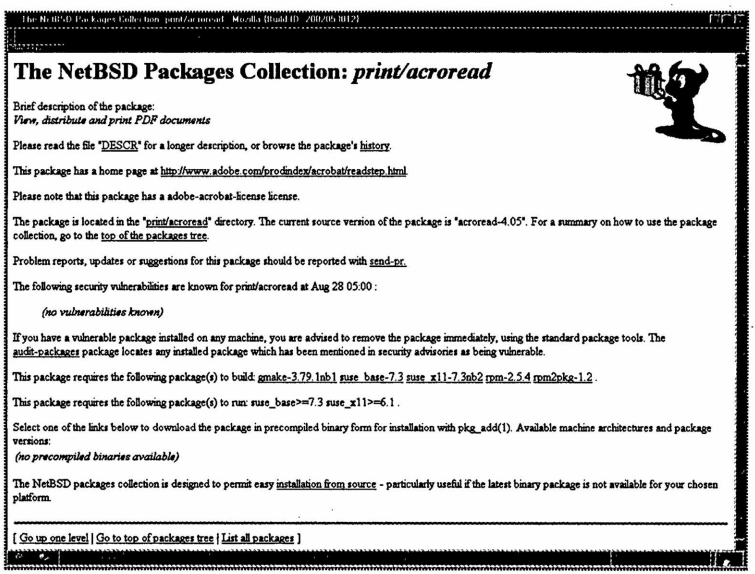


Abb. 5.12: Pakete mit Hinweisen wie »no precompiled binaries available«, die dazu auch noch Emulationsbibliotheken verwenden, sind selten plattformunabhängig

6 Das Betriebssystem aktualisieren

von Dr. Lex Wennmacher

6.1 Das Standardverfahren

Wie jedes andere Betriebssystem sollte man auch NetBSD von Zeit zu Zeit aktualisieren, insbesondere dann, wenn Sicherheitslücken bekannt geworden sind, die es zu beheben gilt. In diesem Kapitel wird beispielhaft die Aktualisierung von NetBSD Serie 1.5, deren letztes Release 1.5.3 war, besprochen.

Das im folgende erläuterte Standardverfahren zur Aktualisierung des Betriebssystems wird vom NetBSD-Projekt empfohlen, es ist auch dokumentiert. Das Verfahren läuft ähnlich wie die Installationsprozedur ab (siehe Kapitel 2). Es wird der Install-Kernel mit dem darin enthaltenen Programm *sysinst* benutzt. Das Standardverfahren fällt aber je nach Rechnerarchitektur unterschiedlich aus, da auf verschiedenen Plattformen beispielsweise verschiedenen Bootmedien zum Einsatz kommen. Einige Architekturen verwenden Floppy-Disketten, bei anderen muß man das Boot-Image in den lokalen Swap-Bereich holen, bei wiederum anderen muß man via Netzwerk booten. Im folgenden wird das Aktualisierungsverfahren für die (am weitesten verbreitete) i386-Architektur beschrieben, also das für PCs. Die Anleitung für die anderen Architekturen ist im jeweils architekturrelevanten *INSTALL*-Dokument beschrieben.

Zunächst muß der Administrator wissen, welche Systeme mit dem Standardverfahren aktualisiert werden können. Das Standardverfahren führt die Aktualisierung eines NetBSD-1.5.3-Systems oder einer früheren Betriebssystemversion durch.

Entwickler-Versionen wie beispielsweise NetBSD-1.6A, 1.6B und so weiter stellen keine »Releases« dar, sondern kennzeichnen Änderungen in den Kernel-APIs. Der NetBSD-1.6-Install-Kernel sollte nicht verwendet werden, um eine Entwickler-Version zu aktualisieren. Dabei kann es zu (zum Teil subtilen) Fehlern kommen.

6.1.1 Vorbereitungen

Backup

Da das Aktualisieren des Systems beispielsweise auch die Bootblocks aktualisiert, besteht natürlich die Gefahr eines kompletten Datenverlusts. Das Anfertigen eines Backups ist daher in den Vorbereitungsarbeiten dringend anzuraten. Zumindest sollten alle Benutzer-Daten gesichert werden sowie die Konfigurationsdateien in »/etc«.

Medien

Ähnlich wie beim Installieren des neuen Systems muß man einen Satz Bootdisketten erstellen. Für NetBSD-1.6 werden zwei Bootdisketten benötigt, die gleichen sind auch für die Installation erforderlich sind. Ich verweise für deren Erstellung auf Kapitel 2 in diesem Buch.

Außerdem werden wie bei der Installation die sogenannten »Sets«, also »base.tgz«, »kern.tgz«, »comp.tgz« und so weiter benötigt. Es gibt eine Reihe von Möglichkeiten, diese Dateien für den Aktualisierungsvorgang zur Verfügung zu stellen: auf dem lokalen Dateisystem, auf einem ftp-Server, einem NFS-Server, auf CD-ROM. Falls möglich, sollte man versuchen, die Sets lokal zur Verfügung zu haben, das heißt, auf einer lokalen Platte oder auf CD-ROM. Das erspart das Konfigurieren des Netzwerks während des Aktualisierungsprozesses, was eine fehleranfällige Prozedur darstellt.

Eigentliche Aktualisierung

Die Durchführung der Aktualisierung geschieht völlig analog zur Installation des Systems, mit dem Unterschied, daß natürlich keine Partitionierung der Platten erfolgt, sondern die bereits bestehenden Partitionen nach einem vorangehendem Lauf von *fsck* verwendet werden. Das Original-Verzeichnis »/etc« wird in »/etc.old« umbenannt und mit Ausnahme der Datei »/etc/fstab« wird kein Versuch unternommen, die neuen Dateien in »/etc« an die in »/etc.old« anzupassen.

Als letzter Schritt der Aktualisierung wird ein neuer Kernel nach »/netbsd« installiert und das System ist ein komplettes NetBSD-1.6-System.

Nacharbeiten

Neue NetBSD-Versionen bringen üblicherweise neue Gerätedateien mit sich. Man sollte daher sicherstellen, daß alle neuen Gerätedateien in »/dev« auch wirklich vorhanden sind:

```
# cd /dev
# sh MAKEDEV all
```

Eine Reihe von Programmen wurden im NetBSD-1.6 Release gegenüber Version 1.5 entfernt, da sie nicht mehr benötigt werden; man sollte sie von Hand löschen (die Aktualisierungsprozedur kümmert sich nicht darum). Üblicherweise erledigt man das, indem man nacheinander in die Verzeichnisse »/bin«, »/sbin«, »/usr/bin«, »/usr/sbin«, »/usr/libexec« wechselt und sich mit

```
# ls -alt
```

die Zeitstempel der Betriebssystemprogramme ansieht. Solche mit einem wesentlich älteren Zeitstempel als die Mehrheit können gelöscht werden. Es sei hier daran erinnert, daß man keine Programme »von Hand« in die oben aufgezählten Verzeichnisse installieren soll, dafür ist der Verzeichnisbaum »/usr/local« mit Verzeichnissen wie »/usr/local/bin« und »/usr/local/sbin« vorgesehen.

Einige Dateien in »/etc« (vor allem solche, die keine Konfigurationsdateien sind) haben seit NetBSD-1.5 Änderungen erfahren, beispielsweise:

- /etc/daily
- /etc/weekly
- /etc/monthly
- /etc/security
- /etc/rc.subr
- /etc/rc
- /etc/rc.shutdown

und weiter. Andere Dateien sind nun obsolet, beispielsweise

- /etc/rc.d/NETWORK
- /etc/rc.d/gated

Des weiteren erwartet NetBSD-1.6 Benutzer und Gruppen für *named*, *ntpd* und *sshd*. NetBSD-1.6 bietet zum automatischen Anpassen dieser Dateien das Werkzeug *postinstall* an.

postinstall operiert auf zwei »etc«-Verzeichnissen, dem alten und – als Vorlage – dem neuen. *postinstall* kennt die notwendigen Anpassungen und versucht, soviel wie möglich von der alten Konfiguration zu bewahren. Es wird folgendermaßen benutzt:

```
# mkdir /tmp/upgrade
# cd /tmp/upgrade
# tar xpzf /Pfad/zu/etc.tgz
# ./etc/postinstall -s `pwd` check
# ./etc/postinstall -s `pwd` fix
```

Als letzten Schritt sollte man noch die Konfigurations- und Startup-Dateien auf den neuesten Stand bringen. Dafür gibt es ebenfalls ein neues Tool: *etcupdate*. Es funktioniert ähnlich wie *postinstall*, kümmert sich aber vor allem um (möglicherweise vom Administrator veränderte) Konfigurationsdateien. So wird es benutzt:

```
# etcupdate -p less -b /tmp/upgrade
```

In der Folge vergleicht *etcupdate* jeweils die alten Konfigurationsdateien im »alten« »/etc« mit den neuen und fragt in jedem Einzelfall den Administrator, welche Aktion erfolgen soll: neue Datei installieren, Dateien mischen, oder Datei löschen.

Damit die erfolgten Änderungen in »/etc« auch alle aktiv werden, empfiehlt sich zum Abschluß noch ein letzter Reboot. Es ist nach einem Upgrade natürlich auch sinnvoll, einen neuen, angepaßten Kernel zu erzeugen. Dies geht besonders einfach mit Hilfe das Pakets *adjustkernel* aus dem Paketsystem (siehe Kapitel 6.2).

6.2 Das Alternativverfahren

Das im folgenden geschilderte Alternativverfahren hat gegenüber dem Standardverfahren den wesentlichen Vorteil, daß es ohne die Verwendung des Installationskernels auskommt. Man bereitet mit dem vollständig konfigurierten alten System die Installation vor und hat dabei alle Dienste wie beispielsweise das Netzwerk zur Verfügung. Man kommt bei der Installation mit einem oder höchstens zwei Reboots aus. Im Prinzip nimmt man beim Alternativverfahren alle Upgradeschritte des Standardverfahrens manuell vor.

Ein weiterer Vorteil des Alternativverfahrens ist, daß es für alle Architekturen gleich ist, da man auf das Booten von einem Installationsmedium verzichten kann.

Die Gefahr beim Alternativverfahren ist, daß man einen notwendigen Schritt vergißt, um den sich das Standardverfahren selbständig kümmert. Daher kann das Alternativverfahren nur erfahrenen Administratoren empfohlen werden, die die Details des Upgradeverfahrens genau verstehen. Die groben Schritte des Alternativverfahrens sind wie folgt:

- Ein sicheres Backup anlegen.
- Ein neues NetBSD-Release (beispielsweise 1.6) auf einen lokalen Datenträger laden.
- Falls erforderlich, die Boot-Blocks aktualisieren.
- Einen neuen Kernel (1.6) bauen.
- Vom neuen Kernel in den Single-User-Modus booten.
- Das neue Release auspacken (bis auf das »etc«-Set).
- *etcupgrade* laufen lassen.
- Falls erforderlich, Dateisystem-Konversionen durchführen.
- Neu booten.

Nun die Erläuterung der Schritte im einzelnen:

Grundsätzlich muß vor dem Upgrade ein komplettes Backup der Maschine, zumindest aber der User-Daten und aller Konfigurationsdateien angefertigt werden. Obwohl beim Upgrade des Systems die User-Daten nicht angefaßt werden, ist das Potential für deren Zerstörung immer gegeben. Zusätzlich müssen einige Dateien gerettet werden:

```
# cp /usr/X11R6/lib/X11/XF86Config /etc
# cp /usr/src/arch/i386/conf/<MYMACHINE> /etc
```

<MYMACHINE> ist im konkreten Fall durch die Kernelkonfigurationsdatei der Maschine zu ersetzen. Jetzt muß »/usr/src« gelöscht werden:

```
# rm -rf /usr/src
```

Dann wird die NetBSD-1.6-Distribution in ein lokales Verzeichnis geholt. Dies geht beispielsweise so:

```
# cd /usr
# mkdir NetBSD-1.6
# ftp ftp.netbsd.org
ftp> bin
ftp> prompt
ftp> cd /pub/NetBSD/NetBSD-1.6/source/sets
ftp> mget *.tgz
ftp> cd ../../i386/binary/sets
ftp> mget *.tgz
ftp> cd ../kernel
ftp> mget *.gz
ftp> exit
```

Im nächsten Schritt werden die Quellen ausgepackt:

```
# mkdir /usr/src
# cat gnusrc.tgz | gzip -c -d | (cd /; tar xvpf -)
# cat sharesrc.tgz | gzip -c -d | (cd /; tar xvpf -)
# cat src.tgz | gzip -c -d | (cd /; tar xvpf -)
# cat syssrc.tgz | gzip -c -d | (cd /; tar xvpf -)
# cat pkgsrc.tgz | gzip -c -d | (cd /usr/pkgsrc; tar xvpf -)
# cat xsrc.tgz | gzip -c -d | (cd /usr/src; tar xvpf -)
```

Und anschließend wird der neue Kernel installiert:

```
# mv /netbsd /netbsd.old
# cat netbsd-GENERIC.gz | gzip -c -d > /netbsd
```

Jetzt muß das System vom neuen Kernel booten:

```
# shutdown -r now
```

Dabei wird in den Single User Mode gebootet:

```
> boot -s
# fsck -p
# mount -at ffs
```

Im Single User Mode werden alle Sets bis auf »etc.tgz« ausgepackt. Dazu verwenden wir die *--unlink*-Option von *tar*, die die Dateien zuerst löscht, bevor sie erzeugt werden. Der Grund ist, daß man damit auch problemlos Dateien ersetzen kann, die noch geöffnet sind. Als ersten Schritt löscht *tar* die upzudatende Datei. Wenn diese Datei noch geöffnet ist, wird zwar der Verzeichniseintrag gelöscht, aber die Inode wird nicht freigegeben. Das heißt, der Kernel kann die Datei nach wie vor benutzen.

Unter dem nun freigewordenen Namen kann das neue Dienstprogramm installiert werden:

```
# cd /usr/NetBSD-1.6
# cat base.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat comp.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat games.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat man.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat misc.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat text.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat xbase.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat xserver.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat xcomp.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat xcontrib.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat xfont.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
# cat xmisc.tgz | gzip -c -d | (cd /; tar --unlink -xvpf -)
```

Die Dateien in »/etc« werden mit Hilfe von *postinstall* und *etcupdate* angepaßt:

```
# mkdir /tmp/upgrade
# cd /tmp/upgrade
# tar xpzf /Pfad/zu/etc.tgz
# ./etc/postinstall -s `pwd` check
# ./etc/postinstall -s `pwd` fix
# etcupdate -p less -b /tmp/upgrade
```

Als nächstes folgt die Anpassung des Kernels. Dafür bietet sich das Paket *adjustkernel* aus den Package-Sourcen an. Dieses Paket vergleicht die beim Booten gefundenen Geräte mit denen, die in der Kernel-Konfigurationsdatei konfiguriert sind, und trägt nicht erkannte Geräte aus.

```
# cd /usr/src/sys/arch/i386/conf
# adjustkernel -f GENERIC -o MYMACHINE
```

Eine Nachbearbeitung der von *adjustkernel* erzeugten Kernel-Konfigurationsdatei ist in der Regel erforderlich, um beispielsweise die gewünschten *options* einzustellen:

```
# vi MYMACHINE
```

Jetzt wird ein neuer Kernel gebaut (siehe Kapitel 4) und der Rechner erneut gestartet.

Es empfiehlt sich unter Umständen, alle installierte Softwarepakete zu löschen und sie später mit NetBSD1.6 neu zu erstellen. Einige Bibliotheken von NetBSD-1.5.3 und älter enthielten Sicherheitslücken, die sich auf statisch gebundene Programme übertragen haben. Außerdem darf man nicht vergessen, »/etc« zurückzukopieren.

7 Das Netzwerk

7.1 Die Grundlagen

von Markus Illenseer

Die NetBSD-Installation umfaßt die Netzwerk-Basiskonfiguration für eine Standard-Workstation. Diese Konfiguration kann erweitert werden für die vielen Netzwerkfähigkeiten von NetBSD bis hin zur Nutzung von Server-Diensten.

Die Netzwerkfähigkeiten von NetBSD sind wie bei jedem modernen Betriebssystem von Haus aus eingebaut, ohne sie ist ein Arbeiten vermutlich gar nicht mehr denkbar. Im Kernel befinden sich die notwendigen Schichten und Stufen, um über einen Netzwerkadapter ankommende Netzwerkanfragen zu verarbeiten. In der Regel werden heutzutage fast ausschließlich TCP/IP-basierende Applikationen unterstützt – immerhin das Protokoll, das das Internet erst möglich machte. Ein anderes unterstütztes Protokoll ist Atalk (AppleTalk) oder AX.25 (Paket Radio), Novell SPX/IPX ist in Vorbereitung.

Die Anzahl der unterstützten Hardware, die Hardware-Komponente der Netzwerkadapter, ist immens und reicht zurück in die Tage des Thick Yellow Cable von Ethernet, aber auch das in der Industrie noch verwendete Arcnet wird unterstützt. Umsetzer ermöglichen die Einbindung virtueller Adapter, die die serielle oder parallele Schnittstelle als Netzwerkadapter zur Verfügung stellen, um etwa über Modem oder direkte Rechner-Rechner-Verbindung zu kommunizieren.

Die Zahl der Applikationen, die TCP/IP als Übertragungsprotokoll verwenden, ist über die Jahre enorm gewachsen. Die Palette reicht von der Basis der Installations- und Manipulations-Tools über Werkzeuge zur Fehlerüberwachung und letztendlich die modernen Anwendungen wie Mail, Internet Browser oder Chat, die jeder kennt.

NetBSD kennt in der Basis-Installation kein Konfigurationsprogramm, mit dem alle nötigen Einstellungen vorgenommen werden, um beispielsweise den einen oder anderen Internet-Dienst einzurichten. Das Installationstool *sysinst* und das Programm *sushi* dienen der Basis-Konfiguration. Es wird in allen Fällen eine Reihe von Konfigurationsdateien, deren Namen eine sehr lange Tradition unter unixähnlichen Betriebssystemen fortsetzen, verwendet. Parallel möglich ist der Einsatz der einzelnen Dateien oder die Konfiguration über die globale Steuerdatei »/etc/rc.conf«, siehe auch Kapitel 3.

Hinweis: Viele weiterführende Elemente der einzelnen Programme und Funktionen können an dieser Stelle nicht umfassend erklärt werden. Die mitgelieferten Manpages von NetBSD sind zwar in englischer Sprache, was niemanden davon abhalten sollte, zu stöbern und zu experimentieren. Die Angabe *(5)* beispielsweise bei der Dateinennung »resolv.conf« bedeutet, daß die entsprechende Manpage in Sektion 5 der Manpages zu finden ist. Manche Programme oder Funktionalitäten sind über mehrere Sektionen verteilt. Die explizite Angabe von 5 kann beim Aufruf der Manpage erfolgen:

```
%> man 5 resolv.conf
```

Zum Beispiel gibt es für das Tool *route* eine Manpage in Sektion 8 und in Sektion 4; in diesem Kapitel ist die Seite in der Sektion 8 wichtig.

Die folgende Aufstellung betrifft ausschließlich das Protokoll TCP/IP. Die relevanten Konfigurationen erklären die Integration der Netzwerkadapter. Weitere Kapitel im Buch erläutern die Einbindung in lokale und globale Netzwerke.

Gestartet wird das Netzwerk letztendlich durch das Skript */etc/rc.d/network*. Es liest alle Konfigurationsdateien aus und initialisiert die Netzwerkadapter, startet die gewünschten Dienste und baut Verbindung zum Rest der Welt auf:

```
%> /etc/rc.d/network start
```

7.1.1 Die Konfigurationsdateien

Datei	Funktion
/etc/rc.conf	Siehe auch die Manpage von *rc.conf(5)* und Kapitel 3 in diesem Buch. Spezifiziert die Systemdienste, darin eingeschlossen die Netzwerkdienste, die beim Systemstart automatisch gestartet werden.
/etc/hosts	Siehe auch die Manpage von *hosts(5)*. Enthält eine elementare Hostnamen- und IP-Nummern-Zuweisungsliste.
/etc/myname	Enthält den »fully qualified hostname«, also den vollständig im DNS spezifizierten Rechnernamen.
/etc/mygate	Enthält das Default-Gateway als IP-Nummer.
/etc/ifconfig.{IF}	Enthält die Einstellungen des Netzwerkadapters *{IF}*.
/etc/ifaliases	Ein Netzwerkadapter kann über ein Alias auch mehr als eine IP-Nummer zugewiesen bekommen.
/etc/resolv.conf	Siehe auch Manpage von *resolv.conf(5)*. Spezifiziert die Namensauflösung von IP-Nummern von Hostnamen.

Tabelle 7.1: Relevante Netzwerk-Konfigurationsdateien

Beispielkonfigurationen

/etc/rc.conf: Diese Datei enthält die Einträge für den Hostnamen und den Default-Gateway; alternativ auch die Einträge für Netzwerkadapter und IP-Aliase:

```
defaultroute="192.168.0.1"
hostname="desktop.domain.de"
ifconfig_rtk0="192.168.0.2"
```

/etc/hosts: Enthält die absolut notwendigen Hostnamen-zu-IP-Nummern-Zuweisungen, die immer dann nötig sind, wenn es keinen DNS-Dienst gibt, beispielsweise in einem kleinen lokalen Netzwerk:

```
::1                     localhost
127.0.0.1               localhost
192.168.0.2             desktop.domain.de desktop
```

Der Name *localhost* ist reserviert für den aktuell zu konfigurierenden Rechner. Die IP-Nummer 127.0.0.1 ist ebenfalls dafür reserviert. Der Rechnername *desktop* ist in diesem Beispiel der IP-Nummer 192.168.0.2 zugewiesen. Die Tabelle kann alle Rechnernamen oder selbstdefinierte Aliase von Rechnern enthalten, mit denen man auch dann noch arbeiten möchte, wenn es kein funktionierendes DNS-System in lokalen Netzwerk gibt. Der Eintrag »::1« ist für IPv6 – den neuen TCP/IP-Aufsatz – vorbehalten.

/etc/myname: Diese Datei enthält den vollständigen Rechnernamen:

```
desktop.domain.de
```

Dieser Eintrag ist allerdings in der Konfigurationsdatei »/etc/rc.conf« deutlich besser aufgehoben. Dort spezifiziert man den Rechnernamen mit *hostname*.

/etc/mygate: Diese Datei enthält einen Eintrag für das Default-Gateway des lokalen Netzwerks:

```
192.168.0.1
```

Auch dieser Eintrag ist in der Datei »/etc/rc.conf« besser untergebracht. Dort wird er mit dem Eintrag *defaultroute* definiert. Alternativ können Sie auch den *routed(8)*-Dienst starten, der vor allem für NetBSD-Server zu empfehlen ist.

/etc/ifconfig.{*IF*}: Wird durch das Netzwerkstart-Skript */etc/rc.d/network* bei Systemstart ausgelesen, um die Netzwerkadapter zu initialisieren und zu konfigurieren. Siehe auch Manpage *ifconfig.if(5)*. Alternativ können die Adapter auch in der Datei »/etc/rc.conf« konfiguriert werden. *IF* ist eine Zeichenkette, die der Kernel einkompiliert bekommt, um einen Netzwerkadapter anzusprechen.

Beispielhafte Dateinamen inklusive des {*IF*} sind:

```
ifconfig.rtk0
ifconfig.epic0
```

Das Suffix *rtk0* und *epic0* entspricht den im Kernel konfigurierten Netzwerkadapter-Namen. Siehe auch Kapitel 3.

Der Inhalt der Dateien lautet wie folgt:

```
inet 192.168.0.2 netmask 255.255.255.0
inet 192.168.0.3 netmask 255.255.255.0 alias
```

Der erste Eintrag bestimmt das Protokoll. Der zweite Eintrag bedeutet, daß der Rechner auch die IP-Nummer 192.168.0.3 zugewiesen bekommt. Dies kann für bestimmte Serverdienste interessant werden. Aus historischen Gründen ist es immer noch sinnvoll, die Adapter über diese Dateien zu konfigurieren statt über die Datei »/etc/rc.conf«.

/etc/ifaliases: Werden in dieser Datei IP-Aliase eines Netzwerkadapters spezifiziert, werden sie von »/etc/rc.d/network« bei Systemstart ausgelesen und initialisiert. Auch dieser Eintrag kann in der Datei »/etc/rc.conf« vorgenommen werden, vorzuziehen ist allerdings die Variante, in der die Aliase in der entsprechenden »ifconfig.{IF}«-Datei eingetragen werden, da dort mehrere Aliase definiert zugeordnet werden können und man so alles bei einem Adapter stehen hat. Siehe auch Manpage zu *ifaliases(5)*.

In dieser Datei sind die IP-Aliase wie folgt hinterlegt:

```
192.168.0.3 rtk0 255.255.255.0
```

/etc/resolv.conf: Hier wird der Name-Server eingetragen, bei dem alle Netzwerk-Programme eine Zuordnung IP-Nummer und Rechnernamen aufzulösen versuchen.

```
nameserver 192.168.0.1
nameserver 68.0.3.1
domain domainname.de
search  domainname.de
lookup file bind
```

Mit dieser Konfiguration wird angegeben, daß der erste Name-Server mit der IP-Nummer 192.168.0.1 (also ein lokaler Server) zuerst zu befragen ist; natürlich muß dieser Server auch einen DNS-Dienst anbieten. Danach ein Server mit der IP 68.0.3.1 – ein fiktives Beispiel für einen Internet-Name-Server Ihres Internet-Providers, den Sie erfragen müssen. Weiterhin sollen Rechnernamen ohne die Angabe der Domain automatisch mit dem Domainnamen *domainname.de* ergänzt werden beziehungsweise in dieser Domain gesucht werden. Die Reihenfolge in *lookup* gibt an, daß zuerst in der Datei (*file*) »/etc/hosts« und dann im DNS (*bind*) auf dem Name-Server zu suchen ist. So kann man sich lokale Aliase für Rechnernamen hinterlegen.

Netzwerkadapter: Das sind physikalische Geräte wie etwa Ethernet-Netzwerkkarten für den PCI-Bus oder PCMCIA-Einschubkarten, oder logische Geräte, die über Basis-Schnittstellen wie die serielle Schnittstelle eingerichtet sind. Die Netzwerkadapter müssen durch den Kernel zur Verfügung gestellt sein, damit man sie einrichten und benutzen kann.

Mit *ifconfig* erhält man eine Liste der aktiven Netzwerkadapter:

```
%> ifconfig -l
epic0 lo0 ppp0 ppp1 sl0 sl1
```

Die Netzwerkadapter *epic0* (Ethernetkarte), *lo0* (lokaler Loopback-Adapter, eine Art Dummy-Netzwerkadapter), *ppp0* und *ppp1* (Point-to-Point-Protokoll für Modem über serielle Schnittstelle) und *sl0* und *sl1* (Slip-Adapter über serielle Schnittstelle, ebenfalls für Modem) sind in diesem Beispiel eingerichtet und verfügbar.

Wird Ihre Netzwerkkarte dabei nicht angezeigt, müssen Sie eventuell einen neuen Kernel kompilieren oder Kernelparameter einstellen. Zum jeweils unterstützten Netzwerkadapter gibt es eine entsprechende Manpage: *epic(4)*, *ppp(4)*, *lo(4)*, *sl(4)* für die Adapter aus obiger Auflistung. Dort finden Sie auch eventuell nötige Parameter zur Ansteuerung beispielsweise eines bestimmten Medien-Typs (BNC-Kabel, Twisted-Pair-Verkabelung und ähnliches). Die Auflistung der unterstützten Netzwerkadapter findet man online bei *http://www.netbsd.org/*.

7.1.2 Einbindung eines Rechners in ein lokales Netz

Will man einen Rechner unter NetBSD ohne komplexe Konfiguration »mal schnell« in ein Netzwerk einbinden, braucht man dazu nur die Programme *ifconfig* und *route* und man muß den Netzwerkadapter kennen. Angenommen, der Rechner soll die IP-Nummer 192.168.0.2 erhalten und sein Default-Gateway ist 192.168.0.1:

```
%> ifconfig lo0 127.0.0.1
%> fconfig ex0 192.168.0.2 netmask 255.255.255.0
%> route add default 192.168.0.1
```

7.1.3 Die Wartung

Programm	Kateg.	Funktion
ifconfig	8	Konfiguration und Anzeige der Daten der Netzwerkadapter.
route	8	Manipulation der Routing-Tabelle.
ping	8	Sendet *ICMP ECHO_REQUEST*-Pakete (eine Aufforderung zur Antwort) an andere Netzwerkrechner.
traceroute	8	Anzeigen des Routings von Paketen zu einem Netzwerkrechner.
tcpdump	1	Anzeigen des Verkehrs auf einem Netzwerkadapter.
netstat	1	Zeigt den aktuellen Netzwerkstatus an.
sstat	8	*systat netstat* zeigt die aktiven Netzwerkverbindungen dynamisch an.

Tabelle 7.2: Überwachungsprogramme und nützliche Netzwerkprogramme

Basis-Sicherheit

Die Dateien »/etc/hosts.allow(5)« und »/etc/hosts.deny(5)« erlauben eine (recht grobe) Einstellung, wer auf Ihrem System per Netzwerk auf Dienste zugreifen darf, die der Rechner zur Verfügung stellt.

Eine Standardregel wäre beispielsweise »Kein Zugang für niemanden«. Das ist schnell erreicht, wenn man die Datei »/etc/hosts.deny« wie folgt einstellt:

```
/etc/hosts.deny:
        ALL: ALL
```

Dies verbietet den Zugang zu allen Diensten von allen Rechnern, es sei denn, dies wird mit Regeln in der Datei »/etc/hosts.allow« explizit erlaubt.

```
/etc/hosts.allow:
        ALL: .domain.de EXCEPT windowspc.domain.de
```

Diese Regel erlaubt allen Rechnern aus der Domain *domain.de* (beachten Sie bitte den führenden Punkt) den Zugang, verbietet ihn dabei weiterhin aber dem Rechner *windowspc.domain.de*. Die beiden Dateien lassen sich auch detailliert für bestimmte Serverdienste einschränken. Mehr Sicherheit gibt es mit den *ipf(8)*-Filtern.

7.1.4 Probleme mit dem Netzwerk

Nichterreichbarkeit anderer Hostnamen per *ping*:

Wenn Sie andere Rechnernamen mit ihrer IP-Nummer (*ping 192.168.0.1*) erreichen können, nicht aber mit dem Rechnernamen (*ping servername*), liegt vermutlich ein Fehler in der Datei »/etc/resolv.conf« vor. Prüfen Sie die eingetragenen Name-Server und die Reihenfolge der Anfragen durch *lookup*.

Verbindungsaufbau hat eine Verzögerung von 30 Sekunden:

In der Regel ist auch dies ein Namensauflösungsproblem. Prüfen Sie die Einträge in der Datei »/etc/resolv.conf« und »/etc/hosts«, ob fehlerhaft konfigurierte Name-Server oder Rechner, die nicht antworten, diese Fehler hervorrufen.

7.1.5 inetd, der Internet Super Server

Dieser »Kontrollserver für Inetnetdienste« wurde vor seit Urzeiten (frühe achtziger Jahre) in unixähnliche Systeme aufgenommen. Wir geben im folgenden eine einfache Definition des *inetd* und zeigen, wie die verschiedenen Konfigurationsdateien mit ihm in Zusammenhang stehen, listen auf, wie man eigene Dienste in den *inetd* aufnimmt und entscheidet, wann man einen Dienst in den *inetd* aufnimmt, oder wann es besser ist, den Dienst ohne *inetd* zu starten.

Der *inetd* lauscht auf seinen eigenen Sockets (Verbindungseinheiten) nach Verbindungsanfragen von außerhalb. Wird eine solche entdeckt, findet er heraus, welcher Dienst anzusprechen ist und startet eine Instanz dieses Dienstes.

Die folgende Grafik zeigt die Abfolge: Der *inetd* erhält Anfragen aus dem Netz beispielsweise an den POP3-Dienst (Mail), ftp- oder den CVSP-Dienst (Control Version System). Der *inetd* multiplext nun die Software und startet automatisch die entsprechenden Dienste, das heißt, er filtert eingehende Verbindungsanfragen und startet (und stoppt) die Dienstprogramme erst bei eingehenden Verbindungen. Damit spart er wertvolle Ressourcen vor allem dann, wenn bestimmte Dienste nicht ständig benötigt werden.

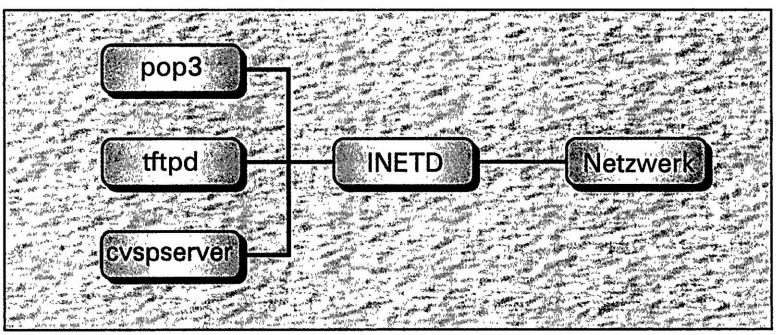


Abb. 7.1: Der inetd ist die Schnittstelle zwischen den Daemons und dem Netz

Der *inetd* – das sei explizit erwähnt – bedient sich einer speziellen Bibliothek *libwrap*, um mehr Sicherheit bezüglich der Verbindungsaufbauten zu gewährleisten.

Die Konfigurationsdateien, Protokolle und Dienste des *inetd* werden im folgenden erläutert.

In der Datei »/etc/protocols« sind die Protokollnamen hinterlegt. Das Format ist dabei sehr einfach gehalten:

```
Protokollname    Nummer      Aliase
```

Die folgende Zeile ist in der Regel der zweite Eintrag in dieser Datei:

```
icmp    1       ICMP            # internet control message protocol
```

Von links nach rechts gelesen sind der Protokollname *icmp*, die Nummer *1* und das Alias *ICMP* spezifiziert. Ab dem »#« beginnt ein Kommentar bis zum Ende der Zeile.

Die nächste Datei »/etc/services« umfaßt die Serverdienste. Diese Datenbank enthält die Informationen über Dienste und ihre Zuweisung von Protokoll zur Port-Nummer. Das Format der Datei sieht wie folgt aus:

```
Dienstname   Portnummer   Protokollname   Aliase
```

Schauen wir uns folgendes Beispiel an:

```
ssh             22/tcp          # Secure Shell
ssh             22/udp
```

Der Dienstname ist also *ssh*, die verwendete Portnummer ist *22*, die verwendeten Protokolle sind sowohl *tcp* als auch *udp*. Es liegen also getrennte Einträge pro Protokoll und Portnummer vor.

RPC (Remote Procedure Call): RPC startet Programme und Dienste auf entfernten Netzwerkrechnern. Die RPC-Programmnummern werden in der Datei »/etc/rpc« hinterlegt, sie enthält eine Zuweisung von RPC-Programmnummern.

```
Dienstname Programmnummer Aliase
```

Schauen wir uns dies am Beispiel von NFS (Network File System) an:

```
nfs             100003  nfsprog
```

nfs ist der Dienstname, die Programmnummer ist *100003* und das Alias ist *nfsprog*.

inetd: Die letzte Datei ist die eigentliche Haupt-Konfiguration des *inetd*, »/etc/inetd.conf«. Hier werden die einzelnen Dienste eingestellt und die Zuweisungen für die Dienste definiert, die der Administrator mit Hilfe des *inetd* gemultiplext haben möchte. Sie ist in der Regel nach einer Neu-Installation von NetBSD zwar gefüllt, aber kein Dienst ist aktiviert, da alle Zeilen mit einem »#« beginnen (das Zeichen für einen Kommentar).

Trotz dem, daß die vorherigen Dateien einfach aussahen, verhält es sich mit dieser Datei leider ganz anders. So sieht eine Zeile der inetd.conf-Datei aus:

```
Dienstname
Sockettyp
Protokoll
Warten/Nicht warten  (wait/nowait)
Benutzer:Gruppe  (user:group)
Dienstprogramm
Dienstprogramm und Argumente für das Dienstprogramm
```

Der Dienstname sollte mit dem Eintrag in »/etc/services« übereinstimmen.

Der Sockettyp (socket type) spezifiziert die Kommunikationsart: *stream, dgram, raw, rdm* und *seqpacket*. Dies ist abhängig von der Funktionsart des Diensts. Die gebräuchlichsten Typen sind *stream* und *dgram*.

Das verwendete Protokoll ist hauptsächlich eine Wahl aus *tcp, tcp6, udp* und *udp6*.

Das Feld *wait/nowait* ist abhängig vom verwendeten Sockettyp. Es definiert, ob der Dienst nach dem erfolgreichen Abarbeiten eine Verbindung sofort schließen oder sie noch aufrecht erhalten soll.

Das Feld *Benutzer:Gruppe* ist selbsterklärend. Der angegebenen Benutzer und die optionale Gruppe werden zur Ausführung des Dienstes verwendet.

Das Feld *Dienstprogramm* enthält den vollständigen Pfad zum Dienstprogramm.

Das Feld der Argumente wird an das Dienstprogramm mit übermittelt. Damit können spezifische Optionen übergeben werden.

Es gibt eine ganze Menge an Parametern und Optionen, die angegeben werden können, deshalb hier ein Beispiel aus der Datei »inetd.conf«:

```
ftp         stream  tcp     nowait  root    /usr/libexec/ftpd     ftpd -ll
```

Der Dienstname ist *ftp*, der Sockettyp ist *stream*, das zu verwendende Protokoll ist *tcp*, das Feld *Warten/Nicht Warten* ist auf *nowait* gesetzt (das heißt, nach Abschluß der Verbindung durch den Client wird der Dienst beendet). Der Benutzer, unter dem der Dienst startet ist *root* (eine durchaus diskussionswürdige Einstellung). Der Pfad zum Dienstprogramm ist */usr/libexec/ftpd* und *Dienstprogramm und Argumente* ist *ftpd -ll*. Beachten Sie, daß im letzten Feld der Dienstprogrammname *ftpd* nicht identisch ist zum Dienstnamen *ftp*.

7.1.6 Einen Dienst hinzufügen

Es ist recht einfach, einen neuen Dienst hinzuzufügen. Da ein Administrator recht häufig in die Verlegenheit kommen wird, einen Dienst hinzuzufügen, beispielsweise weil er noch nicht spezifiziert ist oder weil ein Dienst so wenig Netzwerkverkehr erzeugt, daß es sinnvoll ist, diesen im *inetd* einzufügen. Als Beispiel nehmen wir den Eintrag für *pop3* auf einem NetBSD-System.

Für diesen Mail-Daemon, den man von einem anderen Rechner aus beispielsweise mit einem Mail-Client wie *Pegasus* (Windows) oder *pine* und *mutt* (Unix) anspricht, muß das NetBSD-Package *imap-uw* oder *qpopper* installiert sein. Das POP3-Protokoll ist ein Datemstrom, also *stream* und daher auch mit *nowait* zu bedienen. Den Benutzer *root* zu verwenden, ist dabei vertretbar.

Beim Package *imap-uw* sieht die Konfiguration so aus:

```
pop3    stream  tcp     nowait  root    /usr/pkg/libexec/ipop3d ipop3d
```

Das Package hat das Dienstprogramm */usr/pkg/libexec/ipop3d* installiert, das ohne weitere Parameter funktioniert.

Für das Package *qpopper* sieht es so aus:

```
pop3   stream  tcp     nowait  root    /usr/pkg/libexec/qpopper qpopper -s
```

Der Qpopper benötigt die Option *-s*.

Nach erfolgtem Eintrag muß nur der *inetd* neu gestartet werden, wofür sich das Startskript am besten eignet:

```
%> /etc/rc.d/inetd restart
```

In der Regel dokumentiert Software, die Sie installieren, auch die Einträge für den *inetd*.

Die Entscheidung, ob ein Dienst mittels *inetd* gestartet werden soll oder nicht, müssen Sie anhand der Serverlast selbst entscheiden. Ein Maildienst erfordert vielleicht häufiger neue Verbindungsanfragen als ein *telnet-* oder ein *ftp*-Dienst. In einem solchen Fall wird der Mailserver besser direkt selber gestartet, der *telnet-* und der *ftp*-Dienst durch den *inetd* aber erst explizit durch Verbindungsanfragen.

Der *inetd* stoppt diese beiden Dienste auch, nachdem der Client die Verbindung geschlossen hat. Das spart Speicher und Ressourcen. Andererseits dauert es eine gewisse Zeit, bis ein neuer Prozeß des *ftp*-Diensts gestartet ist. Das kann sich empfindlich bemerkbar machen. Ein schlecht eingestellter Mail-Client, der jede Minute neue Mail auf dem POP3-Dienst abfragt, kann also über den *inetd* unnötig viel Ressourcen verbrauchen. Hier ist es also ratsam, den POP3-Dienst aus dem *inetd* zu entfernen und ihn in »/etc/rc.conf« wieder direkt zu starten – und das, obwohl es sich unter Umständen vielleicht nur um einen Benutzer von insgesamt fünf handelt, der seinen Client so einstellt.

7.2 Die Grundlagen von TCP/IP

7.2.1 TCP/IP v4

von Markus Illenseer

Das um 1979 herum entstandene OSI-Schichtenmodell galt bis weit in die 1990-er Jahre als der Standard für die Beschreibung der verschiedenen Stufen, die bei einer Kommunikation zwischen Computer und Computer notwendigerweise durchschritten werden sollten. Damals wurden sieben Schichten definiert. Fünf davon werden heute auf Millionen von Computern in der Praxis verwendet und begründen das Schichtenmodell, in dem TCP/IP die tragende Rolle spielt. Am Beispiel einer Verbindung zweier Computer auf Basis von Ethernet wollen wir die Schichten durchleuchten.

1. Physikalische Schicht Ethernet

Diese Schicht besitzt die direkte Kontrolle und den Kontakt zum physikalischen Medium, das zur Kommunikation gewählt wurde. Die elektrischen Signale zwischen den beiden Ethernetkarten in beiden Rechnern enthalten vorerst kaum mehr als ein paar Bits. Die Treibersoftware unter NetBSD sorgt für den Betrieb und vor allem für die Erkennung der Hardware. Wie Ethernet (als Hardware-Kommunikationsprotokoll) generell funktioniert, müssen wir hier leider außen vor lassen, da dies den Rahmen sprengt.

2. Internet Protocol (IP)

Paketorientierte Protokolle wie IP regeln in dieser Schicht den Verbindungsauf- und abbau. Dies ermöglicht durch die Steuerung der Wegfindung den Aufbau von Netzwerkknoten und strukturierten Netzwerken. IP verpackt die zu übertragenden Daten in handliche Pakete, die einen Absender und einen Empfänger enthalten, und reicht sie an die Ethernet-Schicht weiter. Auf dem Empfänger-Computer erhält IP das Paket von der Ethernet-Schicht und packt es wieder aus. Damit können wir nun schon ganze Pakete hin- und herschicken. Da Ethernet auf Grund seiner Funktionsweise aber nicht die Reihenfolge (und andere Faktoren) der Pakete garantieren kann, brauchen wir einen weiteren Kontrollmechanismus.

3. Transmission Control Protocol (TCP)

Hier werden die von IP empfangen Daten kontrolliert, sortiert und in die richtige Reihenfolge gebracht, es werden auch defekte Pakete vom Absender erneut angefordert.

Die beiden Schichten IP und TCP werden oft zusammengefaßt als TCP/IP Protocol Stack bezeichnet. Der Stack alleine ist sinnlos, wenn es keine Applikationen gibt, die ihn nutzen können. Damit diese loslegen können, wurden Dienste definiert, mit deren Hilfe die Applikationen und damit der Anwender arbeiten können.

4. Dienstprotokolle

Es wurden seit der »Erfindung« der TCP/IP-basierten Netzwerke verschiedene Dienste entwickelt. Einige gibt es seit Anfang an, andere wurden inzwischen längst wieder verschwunden und durch neue Dienste abgelöst.

Für den Dienst E-Mail gibt es das Protokoll »Simple Mail Transfer Protocol« (SMTP), das die Übertragung von E-Mail auf Basis von TCP/IP regelt. Weitere bekannte Dienste sind »Net News Transfer Protocol« (NNTP) und »File Transfer Protocol« (*ftp*). Das mittlerweile bekannteste Protokoll ist das »Hyper Text Transfer Protocol« (HTTP), das für die Verwendung und den Betrieb des WWW benötigt wird.

5. Anwendungsschicht

Der Anwender, also Sie und ich, bedienen sich aber lediglich einer Software, die beispielsweise E-Mail mit Hilfe von SMTP austauscht, also eines E-Mail-Clients wie Outlook, pine, mutt, Netscape Messenger und so weiter. Der Client sendet die verfaßte E-

Mail über das Dienstprotokoll SMTP an das Übertragungsprotokoll TCP zur Verbindungsebene IP an die physikalische Schicht Ethernet zum nächsten Rechner. Bei HTTP verwendet man einen Web-Browser, um einen HTTP-Server (beispielsweise Apache) anzusprechen. So gibt es also mindestens eine spezifische Anwendung für ein Dienstprotokoll. Natürlich gibt es auch universelle Tools wie Mozilla, das sowohl als E-Mail-Client, Web-Browser, News-Reader und *ftp*-Client funktioniert.

Im Gegensatz zum eingangs beschriebenen OSI-Referenzmodell, das sich nie wirklich durchsetzen konnte, wird in der Praxis inzwischen vorwiegend das TCP/IP-Schichtenmodell verwendet. Das TCP/IP-Schichtenmodell wurde vor allem in Hinblick auf zwei wichtige Aspekte entwickelt:

Zum einen wurde Wert auf die Fähigkeit gelegt, mehrere Netze zusammenzuschließen – egal, welche Hardware dabei verwendet wird. Aber andererseits soll das Netz trotz Zerstörung oder Lahmlegung von Hardware zumindest in einigen Teilen weiterhin funktionstüchtig bleiben. In der Anfangsphase des Internets war das ein extrem wichtiger Aspekt, inzwischen ist die Erreichbarkeit und der Zusammenschluß der vielen Teilnetze zu großen Teilen wirtschaftlich geregelt.

Da IP für die Datenübertragung zwischen Rechnern zuständig ist, braucht es einen Mechanismus, um die Ziel- und Absendeadressen eindeutig zu definieren. Diese Adressen bestehen aus einer 32 Bit großen Dualzahl. Damit die Strukturierung der Netzwerktopologien einfacher fällt, ist diese Zahl in vier Teile zu je acht Bit zerlegt. Damit diese Dualzahl für den Menschen lesbar ist, werden stattdessen Dezimalzahlen verwendet und die vier verwendeten Zahlen (die 8 Bit) durch Punkte getrennt. Als Beispiel kann eine solche IP-Nummer so aussehen: 192.168.0.1 oder 10.10.20.10. Eigentlich ist diese Darstellung fehlerhaft, da sie ja die Berechnung der Dualzahl außer acht läßt. Aus 10.10.20.10 wird 10*256^3+10*256^2+20*256^1 = 168432640. Dies ist eine gültige IP-Adresse – probieren Sie es aus! In dualer Schreibweise ist dies 10100000.10100001.0100000.00000000.

Die IP-Adressen werden in vier Gruppen beziehungsweise Klassen unterteilt. Die nun folgende Nomenklatur nach Klasse-A- bis Klasse C-Netzen ist inzwischen – nach einer internationalen Vereinbarung der Internet Service Provider – eigentlich obsolet. Sie ist aber für den Normalfall nach wie vor gültig und sehr anschaulich.

Klasse-A-Netz

Hier ist das erste Bit im ersten Byte 0, alle anderen Bits im ersten Byte sind beliebig. Gültige Werte des ersten Bytes sind damit zwischen 0 und 127. Damit sind nur 128 Netze des Typs Klasse A möglich. In dezimaler Notation sind dies die Netze zwischen 0.0.0.0 und 127.0.0.0. Die restlichen drei Byte stehen Rechnern zur Verfügung, also 256^3 = 16777216 Rechner. Nur sehr wenigen Institutionen sind solche Netzwerke vorbehalten.

Klasse-B-Netz

Die ersten beiden Bits im ersten Byte lauten 10. Das erste Byte kann damit 64 Werte annehmen. Hier ist das zweite Byte noch Bestandteil der Netzadresse. Daher gibt es

64*256 = 16128 Klasse-B-Netze. In dezimaler Notation sind dies die Netze zwischen 128.0.0.0 und 191.255.0.0. In jedem dieser Netze sind 256^2=65536 Rechner enthalten. Die meisten (international tätigen) Internet Service Provider besitzen mindestens ein solches Klasse-B-Netz, oft auch mehrere.

Klasse-C-Netz

Die ersten drei Bits des ersten Byte lauten hier 110. Damit kann das erste Byte nur noch 32 Werte annehmen. Da laut Definition hier aber das zweite und dritte Byte ebenfalls zum Netzanteil gehören, sind hier nun 32*256*256 = 2097152 verschiedene Netze möglich. Jedes dieser Netze enthält nur 256 Rechner. In dezimaler Notation sind dies die Netze 192.0.0.0 bis 223.255.255.0.

Diese Angaben gelten nur für IP mit der Version 4. Im nächsten Kapitel erfahren Sie, wie IP Version 6 mit der doch inzwischen knapp gewordenen Ressource-IP-Nummer umgeht und diese erheblich erweitert.

Beispiele für Rechneradressen und Netzwerkadressen:

62.28.34.12 ist ein Rechner in einem Klasse-A-Netz. Er gehört zum Netzwerk 62.0.0.0.

190.128.12.18 ist ein Rechner in einem Klasse-B-Netz und gehört zum Netzwerk 190.128.0.0.

201.42.123.62 ist ein Rechner in einem Klasse-C-Netz und gehört zum Netzwerk 201.42.123.0.

Die Netzwerkadresse endet also immer auf eine 0 und ist selber eine IP-Adresse, ohne dabei ein Rechner zu sein. Netzwerkadressen werden im Normalfall zur Wegleitung (Routing) herangezogen und dürfen nicht für Rechner verwendet werden.

Broadcastadressen

Setzt man den Rechneranteil auf 255, erhält man – vereinfacht gesprochen – die Broadcastdresse, mit der man alle Rechner in einem Netzwerk auf einmal erreichen kann.

Mit der Adresse 201.42.123.255 werden alle (aktiven) Rechner im Netzwerk 201.42.123.0 gleichzeitig erreicht.

Netzwerkmaske und Subnetze

Um verschwenderischen Umgang mit Netzwerken Einhalt zu gebieten, kann man IP-Netze auch in logische Teile unterteilen, die man »Subnetz« nennt. Um dieses sehr komplexe Thema nicht unnötig breitzutreten, betrachten wir nur die Netzmaske für ein Klasse-C-Netz. Die Netzmaske ist ein im dualen System auf 1 gesetztes Bit an der Stelle, deren Teil einer IP-Adresse als Netzwerkanteil zu interpretieren ist. Bei einem Klasse-C-Netz ist dies 11111111.11111111.11111111.00000000 beziehungsweise in dezimaler Notation: 255.255.255.0. NetBSD verwendet oft auch die sedezimale Schreibweise, hier also 0xffffff00, um die Maske anzugeben.

Ein Klasse-C-Netz kann man nun mit der Netzmaske 255.255.255.192 (also 256 bis 192 = 64 Rechner pro Subnetz) in vier Subnetze mit jeweils 64 Rechnern unterteilen. Dabei geht das erste Subnetz von 201.42.123.0 bis 201.42.123.63, das zweite Subnetz von 201.42.123.64 bis 201.42.123.127, das dritte Subnetz von 201.42.123.128 bis 201.42.123.191 und das vierte Subnetz von 201.42.123.192 bis 201.42.123.255.

Die jeweils letzte IP-Adresse eines Subnetzes ist die Broadcast-Adresse für dieses Subnetz. Mit der Broadcast-Adresse 201.42.123.63 erreicht man also alle Rechner von 201.42.123.0 bis 201.42.123.62 auf einmal.

Die 64-er Subnetze kann man noch weiter unterteilen. Die kleinsten Subnetze können also minimal von der Größe 4 sein. Da die Netzwerkadresse eine eigene IP-Adresse benötigt, die Broadcast-Adresse ebenso, bleiben zwei IP-Adressen für echte Rechner.

7.2.2 Einführung in IPv6

von Ignatios Souvatzis, Mitglied der IPv6-Arbeitsgruppe der Internet Engineering Taskforce und NetBSD-Entwickler

Seit einigen Jahren sind Bestrebungen im Gange, das alte IP-Protokoll mit der Versionsnummer 4 durch ein neues zu ersetzen, das unter anderem einen erweiterten Adreßraum bereitstellt. Die Kernbereiche des Protokolls – es hat die Protokollversion 6 enthalten, daher der Name IPv6 – sind inzwischen in einer überarbeiteten Version als Draft Standard definiert, es gibt eine Reihe von Implementierungen.

Für die BSD-Systeme hat das japanische KAME-Konsortium[2] eine Referenzimplementierung bereitgestellt, die inzwischen unter anderem in NetBSD integriert ist. In diesem Kapitel soll zunächst das IPv6-Protokoll kurz beschrieben werden, soweit das für die Anwender und Systemadministratoren wichtig ist.

Einordnung im Schichtenmodell

IPv6 ersetzt IPv4 im Schichtenmodell. ICMPv6, das Protokoll, das für die Signalisierung beispielsweise von Fehlerzuständen formal oberhalb von IPv6 benutzt wird, ersetzt das alte IPv4 und hat weitere Aufgaben übernommen, beispielsweise die Adreßauflösung, die für IPv4 parallel zu IP auf der gleichen Ebene stattfindet.

Alle anderen Protokolle – besonders interessant sind heutzutage hauptsächlich TCP und UDP – sind praktisch unverändert geblieben. Probleme sind nur dort zu erwarten, wo Adressen innerhalb der Daten übertragen werden, beispielsweise bei Routingprotokollen oder bei Rendezvous-Protokollen für weitere Verbindungen (*ftp*, Talk, IRC, ...), für die gebräuchlichen gab es auch hier Erweiterungen.

IPv6-Adreßtypen{3}

IPv6-Adressen sind 128 Bit lang. Sie werden, wenn sie textuell notiert werden müssen, in acht Gruppen zu vier Hexziffern notiert (mit Klein- oder Großbuchstaben), zum Beispiel:

```
2003:1234:5678:9ABC:DEF0:C001:BEEF:C001:BEEF
```

Dabei dürfen *je Gruppe* führende Nullen weggelassen werden:

```
2003:0567:5678:9ABC:DEF0:0001:BEEF:0001:BEEF ist das selbe wie
2003:567:5678:9ABC:DEF0:1:BEEF:1:BEEF
```

Eine Gruppe von 0000 am Anfang, in der Mitte oder am Ende darf durch »::« abgekürzt werden, wenn das Resultat eindeutig ist, beispielsweise

```
2003:0567:5678:def0::1
```

bedeutet

```
2003:0567:5678:def0:0:0:0:1
```

beziehungsweise

```
2003:0567:5678:def0:0000:0000:0000:0001
```

Manchmal will man ein Netz und seine Größe bezeichnen. Dies geschieht, ähnlich wie teilweise schon für IPv4 üblich, durch Präfix/Länge (in Bits), beispielsweise ist 3ffe::/16 der Adreßbereich des 6BONE. Die Adresse eines Interfaces und die Größe des Netzwerks, an die es direkt angeschlossen ist, werden genauso durch Adresse/Länge angegeben. Nichtkontinuierliche Netzmasken, wie in IPv4 prinzipiell möglich (wenn auch selten im Einsatz), gibt es in IPv6 mit Absicht nicht. Soweit wie irgend möglich soll nämlich hierarchisch geroutet werden, um die Größe der Routingtabellen zu minimieren.

IPv6 unterscheidet drei Adreßtypen: Unicast-Adressen (ein Knoten sendet an an einen anderen), Multicast-Adressen (ein Knoten sendet an alle Mitglieder einer Empfängergruppe) und Anycast-Adressen (ein Knoten sendet an das netzwerkmäßig nächstgelegene Mitglied eine Gruppe).

Dabei sind Anycast-Adressen bisher nicht von großer Bedeutung, da Anycast-Verkehr mit einigen Protokollproblemen behaftet ist, siehe [4]. In IPv6 – im Gegensatz zum entsprechenden Protokollvorschlag für IPv4 [5] – sind Anycast-Adressen von Unicast-Adressen nicht unterscheidbar.

Broadcast-Adressen – im Sinne von »alle Maschinen auf diesem physikalischen Netzwerk« – sind von IPv6 aus nicht ansprechbar. Alle IPv6-Rechner (genauer: alle IPv6-Interfaces) auf einem lokalen Netzwerk kann man jedoch erreichen, dafür gibt es eine vordefinierte Multicast-Adresse. Protokolle, die in IPv4-Broadcasts benutzten (beispielsweise ARP), wurden durch Protokolle ersetzt, die Multicasts verwenden.

Multicast-Adressen werden wie in IPv4 zum einen für lokale Netzwerkdienste verwendet. Zum Beispiel setzen Routingprotokolle Multicastadressen ein, um beispielsweise alle lokalen Router anzusprechen. Zum anderen sind sie die Methode der Wahl, um offene Konferenzsysteme mit vielen Teilnehmern oder ähnliches zu implementieren.

IPv6-Multicast-Adressen sehen folgendermaßen aus:

Binär:

```
1111 1111 KKKK NNNN ....
```

Hexadezimal:

```
FFKN:....
```

Wobei *N* die Reichweite definiert und *K* einen Satz von Flags, von denen bis jetzt nur das niedrigwertigste definiert ist, die anderen bleiben 0; permanent definierte Multicast-Adressen haben dort eine 0, temporär definierte haben eine 1 (und gelten nur innerhalb des jeweiligen Scopes).

Einige nützliche Beispiele:

ff02::1	Alle Interfaces auf dem lokalen Link.
ff02::2	Alle Router auf dem lokalen Link.

Link-lokale Adressen müssen im KAME-Code durch Anhängen des Interfacenamens (mit »%« getrennt) qualifiziert werden, also beispielsweise:

ping6 ff02::1%ed0	*ping* auf alle IPv6-Rechner am gleichen Netzwerksegment wie mein Interface »ed0«.

Link-lokal, Site-lokal, global: Adreß-Reichweiten (»Scopes«)

IPv6 unterscheidet folgende Adreßreichweiten:

Link-lokal:	Gültig auf einem Link (logischen Netzwerkkabel, z.B. Ethernet).
Site-lokal:	Gültig innerhalb einer Organisation (sofern diese mit 80-Bit-Adreßraum auskommt).
Global:	Alles andere.

Der Scope für Multicast-Adressen ist etwas feiner unterteilt (siehe Tabelle 7.3).

Adresse	Scope
FFt1:...	Node-local, nur innerhalb eines Rechners gültig.
FFt2:...	Link-local, nur auf einem Kabel (-segment).
FFt5:...	Site-local, innerhalb eines Standorts.
FFt8:...	Organization local, innerhalb einer Organisation.
FFtE:...	Global.

Tabelle 7.3: Adreßreichweiten (Scopes) bei IPv6-Multicast. t=0: permanent definierte Adresse, t=1: vorübergehend zugewiesene (transiente) Adresse

Den anderen Multicastpräfixen ist keine Bedeutungen zugewiesen, mit der Ausnahme von FF00:... und FF0F:..., die reserviert sind.

Für *t=0* bleibt die logische Bedeutung unabhängig vom Scope erhalten (sofern überhaupt zugewiesen – *all-nodes* gibt es nur node- und link-lokal, *all-routers* nur node-, link und site-lokal). Man könnte also alle NTP-Server im LAN, alle NTP-Server am Standort oder sogar alle NTP-Server der Organisation nur durch Austausch der vierten Hexziffer ansprechen. Transiente Multicastadressen sind hingegen nur für einen Scope (vorübergehend) reserviert.

Link-lokale Unicast-Adressen fangen mit FE80:0:0:0: an. Auch link-lokale Unicast-Adressen werden durch Anhängen des Interfacenamens qualifiziert, beispielsweise

```
ping6 fe80::a00:2bff:fe18:6305%ed0
```

Site-lokale Unicast-Adressen beginnen mit FC80:0:0:yyyy:. Das *yyyy*, der sogenannte Site Level Aggregate-Identifier (SLA), dient zur Adressierung von bis zu 65536 verschiedenen Netzwerksegmenten innerhalb der Site (eines organisatorischen und netzwerkmäßigen Zusammenhangs von Rechnern, beispielsweise die Rechner bei mir zu Hause oder aber alle Rechner einer Universität).

Pakete mit link-lokalen Zieladressen werden von Routern nie weitergeleitet, Pakete mit site-lokalen Zieladressen werden von Site-Außenroutern daran gehindert, die Site zu verlassen.

Zur Zeit sind drei globale Adreßblöcke in Benutzung:

- 3FFE::/16: Das 6BONE[6], ein experimentelles IPv6-Netz, das überwiegend über IPv4-Verbindungen getunnelt wird.
- 2001::/16: Der erste für Produktionszwecke gedachte Adreßblock.
- 2002::/16: Das sind die 6to4-Adressen, eine Möglichkeit, automatisch ohne Vorkonfiguration Tunnel zwischen Sites über IPv4 herzustellen, siehe unten oder [7]. Dies ist nach dem 6BONE der zweite global definierte Übergangsmechanismus zu IPv6.

Es gibt außerdem zwei Adreßtypen, die mit 0000: anfangen und mit einer IPv4-Adresse enden:

- 6over4, ein Übergangsmechanismus, der keine große Verbreitung gefunden hat:

  ```
  0000:0000:0000:0000:0000:0000:ip.v4.adre.sse
  ```

- IPv4-mapped IPv6: Ein Übergangsmechanismus für IPv6-Software, die auch IPv4 verstehen soll, ohne das IPv4-Socket-Adressenformat (*AF_INET*) zu kennen:

  ```
  0000:0000:0000:0000:0000:ffff:ip.v4.adre.sse
  ```

In NetBSD kann das Erkennen dieser Adressen über den *sysctl*-Schalter

```
net.inet6.ip6.bindv6only=0
```

ein- und mit

```
net.inet6.ip6.bindv6only=1
```

abgeschaltet werden, letzteres ist Normalzustand. Das heißt, daß, wenn in »/etc/inetd.conf« ein tcp6-Dienst aktiviert ist, der Dienst nicht über IPv4 zugänglich ist (wenn nicht der gleiche Dienst auch für tcp aktiviert ist). Dies erleichtert die Zugriffskontrolle mittels »/etc/hosts.allow«.

IPv6 im Distributed Name Service (DNS)

IPv6-Adressen und -Rückwärtspointer werden wie IPv4-Adressen mit der Klasse *IN* notiert. IPv6-Adreßeinträge zu symbolischen Namen erhalten den Typ *AAAA* statt *A*, IPv6-Namen zu Adressen sind wie bei IPv4 *IN PTR*. Allerdings wird hier die Adresse hexzifferweise rückwärts unter *ip6.arpa*, aus historischen Gründen auch noch unter *ip6.int*[8] eingetragen (bei IPv4 byteweise dezimal rückwärts unter *ip.arpa*).

Ab und zu findet man Hinweise auf die »A6«-Records, und »bitstring-labels« für die Rückwärtsauflösung, die zusammen mit DNAMEs (eine Abwandlung von CNAMEs, die auf die Mitte einer Domain zeigen statt auf einen ganzen Namen) entworfen wurden, um das Umnumerieren von IPv6-Teilnetzen im DNS zu unterstützen – diese hätten es erlaubt, ganzen Subnetzen ein neues Präfix zu verleihen, indem man nur einen Eintrag im DNS ändert. Diese Vorschläge blieben jedoch im Status »experimentiell«, weil starke Zweifel an ihrer Praktikabilität auftraten, speziell wenn wie angestrebt DNS-Einträge in Zukunft kryptographisch signiert werden sollen.

Eine Besonderheit ist bei Mailserver-Einträgen zu beachten: Im Gegensatz zu IPv4 ist der AAAA-Eintrag zu einem Domainnamen *nicht* automatisch der zuständige Mailserver, es muß immer einen MX-Eintrag geben, wenn Mail zugestellt werden soll.

AAAA- und PTR-Einträge kann Bind8 (wie in NetBSD integriert) durchaus verwalten, sofern Sie nur über IPv4 abfragen (lassen) wollen. BINDv9 liegt für NetBSD als Package vor [9] und unterstützt Abfrage über IPv6, A6, DNAME und bitstring labels.

Weiteres zur Nameserver-Konfiguration finden Sie weiter weiter hinten in diesem Kapitel.

In »/etc/hosts« können Sie IPv6-Adressen übrigens genauso wie IPv4-Adressen eintragen.

Einige Bemerkungen zu Anwendungen

Anwendungen, die die neuen Standardschnittstellen für Namensauflösung benutzen, können protokollunabhängig geschrieben werden. Die meisten Anwendungen sind inzwischen angepaßt und können »einfach so« auch über IPv6 verwendet werden, beispielsweise telnet und ftp.

Das gilt auch für die RPC- (Remote Procedure Call) Library. Vielleicht sind Ihnen schon die Einträge mit udp6 und tcp6 in »/etc/netconfig« aufgefallen, der Datei, die Transportdienste für den (seit NetBSD 1.5 neuen) »Transport Independent Remote Procedure Call«-Code (TI-RPC) konfiguriert. Sie brauchen nichts daran zu ändern, diese Datei ermöglicht in einem frisch installierten NetBSD-System lokalen Zugriff, sowie UDP und TCP jeweils über IPv4 und IPv6, so daß insbesondere auch das Network Filing System (NFS) über IPv6 benutzt werden kann, client- wie serverseitig.

Im folgenden einige Spezialfälle:

Diagnoseanwendungen

KAME enthält spezielle IPv6-Versionen von *ping* und *traceroute* mit den schönen Namen *ping6* und *traceroute6*. Denken Sie daran, bei link-lokalen Adressen das Interface mit anzugeben, beispielsweise

```
ping6 fe80::34:456%ne0
```

Das ARP-Protokoll zur Adreßumsetzung auf Ethernet wurde für IPv6 durch das Neighbour Discovery Protocol ersetzt. Statt des arp-Programms kann für ähnliche Zwecke in der Regel ndp benutzt werden; zum Beispiel zeigt *ndp -a* den Neighbour Cache an, das Gegenstück zur ARP-Tabelle.

Clientanwendungen

Viele Clientanwendungen sind inzwischen in IPv6-tauglichen Versionen integriert (ssh, telnet, ftp, finger...) oder in der Package-Sammlung erhältlich.

Serveranwendungen

TCP- und UDP-Server, die über *inetd* aufgerufen werden, können genauso gut mit IPv6 arbeiten. Fügen Sie dazu einen zweiten Eintrag in »/etc/inetd.conf« ein, der statt *tcp tcp6*

beziehungsweise anstelle von *udp udp6* enthält. »inetd.conf« umfaßt schon eine Reihe von auskommentierten Beispielen dazu.

Andere Server liegen oft schon in angepaßten Versionen vor, etwa *ssh*. Auch die Ports-Sammlung enthält vieles, was schon IPv6-tauglich ist (etwa den WWW-Cache *wwwoffle*, der als einer der ersten IPv4-Browser den Zugriff auf das IPv6-WWW verschafft hat).

Auch IPv6-Serverdienste können durch »/etc/hosts.allow« geschützt werden, siehe dazu die Beispiele in *man hosts.allow*.

IPv6-untaugliche Dienste

Das sind Dienste, die eine numerische Netzwerkadresse über die Netzwerkverbindung übertragen, etwa der DCC-Teil von IRC und talk. Wenn das jeweilige Protokoll nicht erweitert wird, wie das bei *ftp* geschehen ist, ist nicht damit zu rechnen, daß diese Dienste in die IPv6-Welt hinübergerettet werden können.

Literatur und Verweise

[1] Die Anleitung zu IPv6 auf NetBSD: *http://www.netbsd.org/Documentation/network/ipv6/*

[2] Autoren des IPv6-Codes in (u.a.) NetBSD: *http://www.kame.net/*

[3] IPv6 Addressing Architecture: RFC 2373

[4] Eine Analyse von Anycast von einem der KAME-Entwickler: *http://search.ietf.org/internet-drafts/draft-ietf-ipngwg-ipv6-anycast-analysis-01.txt*

[5] Partridge's Anycast-Vorschlag für IPv4: RFC 1546

[6] *http://www.6bone.net/*

[7] Das Standard-Dokument über 6to4: RFC 3056

[8] Übergang ip6.int nach ip6.arpa: RFC 3152/BCP 52

[9] *ftp://ftp.netbsd.org/pub/NetBSD/packages/pkgsrc/net/bind9/README.html*

7.3 DNS und andere Netz-Konfigurationen

von S. P. Zeidler

7.3.1 Netzwerkkonfiguration als DHCP-Client

Wenn ein Server für das Dynamic Host Configuration Protocol (DHCP) im lokalen Netz arbeitet, setzt man in der Datei »/etc/rc.conf« die Variable *dhclient* auf *YES* und läßt den DHCP-Server die ganze Arbeit tun:

```
dhclient=YES
```

Weitere Einstellungen sind nicht nötig. Das *dhclient*-Skript sucht selbständig den DHCP-Server und erfragt die Adresse.

7.3.2 Statische Netzwerkadressen

IP-Adresse und Netzmaske

Das Eintragen der Werte:

IP-Adresse und Netzmaske werden dem Kernel mit dem Programm *ifconfig* mitgeteilt. Eine Kommandozeile für eine Ethernetkarte könnte zum Beispiel so aussehen:

```
ifconfig de0 inet 10.1.1.1 netmask 255.255.255.240
```

Dabei ist *de0* die erste Ethernetkarte des Typs »de«. Wie die Ethernetkarte(n) im Rechner heißen, kann man beim Booten herausfinden, wenn man die Kernelmeldungen bezüglich aufgefundener Hardware beobachtet, oder danach mit *more /var/run/dmesg.boot* nachsehen. Um diesen Befehl nicht nach jedem Bootvorgang wiederholen zu müssen, legt man eine Datei »/etc/ifconfig.<Interfacename>« an, bei obigem Beispiel »/etc/ifconfig.de0«, und trägt dort das Kommando nach dem Namen des Interfaces ein, also hier *inet 10.1.1.1 netmask 255.255.255.240*. In diese Datei kann man auch weitere Argumente zu *ifconfig* und weitere Zeilen eintragen, um zum Beispiel auch IPv6-Adressen statisch zu konfigurieren.

IP-Adresse und Netzmaske bezieht man vom Netzwerk-Administrator der Installation, in der man sich befindet. Sofern man selbst der Netzwerk-Administrator ist und keine Verbindung ins Internet hat, verwendet man IPv4-Adressen aus folgenden Bereichen: 10.x.x.x, 172.16.x.x bis 172.31.x.x oder 192.168.x.x. Die Netzmaske wählt man dann so, daß die erwartete Menge Rechner bequem hineinpaßt, üblicherweise 255.255.255.0, die Maske mit 254 verwendbaren Adressen. Für IPv6 verwendet man am bequemsten die *autohost*-Konfiguration, die Adressen finden sich dann von selber.

Ist man selbst der Netzwerk-Administrator und hat eine Verbindung zum Internet, kommen der lokale IP-Adreßbereich und die Netzmaske für die Internet-Verbindung vom Provider, bei einfacher Einwahl mit PPP wird die IP-Adresse (speziell bei der bei

Einwahl üblichen dynamischen Adreßvergabe) im PPP-Protokoll übermittelt (und die Netzmaske ist dann 255.255.255.255), für Details siehe Kapitel 8 (Einwahl).

Wozu dient das?

Für eine gelungene Datenkommunikation müssen Nachrichten von einem Rechner zum anderen gelangen. Um zu wissen, wohin mit einer bestimmten Nachricht, braucht diese eine Zieladresse und damit die andere Seite antworten kann, eine Absendeadresse. Diesen Zweck erfüllen bei IP die IP-Adressen; die IP-Adressen in einem verbundenen Netz müssen eindeutig sein (Highlander-Prinzip: Es kann nur einen geben!). Daraus folgt, daß die IP-Adressen zur Nutzung im größten verbundenen Netz, dem Internet, weltweit eindeutig sein müssen. Um diese Eindeutigkeit zu garantieren, wachen sogenannte Address-Registries (Adreß-Vergabestellen) über die IP-Adreßräume im Internet. Die »höchste Stelle« ist dabei derzeit ICANN, die an die »Regional Registries« RIPE (Europa sowie Teile von Afrika), ARIN (Amerika und Teile von Afrika) und APNIC (Asien und Pazifik) Adreßräume zur Verwaltung übergibt, die diese wiederum an weitere Organisationen weitergeben. In der RIPE-Region übergibt RIPE Adreßräume an sogenannte Local Internet Registries, LIRs, die üblicherweise Internet-Provider sind und diese Adressen dann an ihre Kunden verteilen.

Eine IPv4-Adresse besteht aus einer Zahl zwischen 0 und 4294967295 und wird üblicherweise als vierstellige »Zahl« mit »Ziffern« zwischen 0 und 255 dargestellt: so wäre 0 0.0.0.0 und 1 0.0.0.1 und 4294967295 ist 255.255.255.255. Letztere hat übrigens eine Sonderbedeutung, es handelt sich um ein Broadcast (also eine Sendung an alle Teilnehmer) für das lokale Netz. In einem IPv4-Adreßbereich (definiert durch die Netzmaske) ist die größte IP-Adresse immer der Broadcast an alle Rechner in diesem Adreßbereich und kann nicht an einen Rechner vergeben werden. Aufgrund eines historischen Fehlers wird üblicherweise auch die niedrigste Adresse in einem Bereich nicht vergeben, obwohl das sehr wohl möglich ist, wenn man hinreichend Kontrolle darüber hat, welche Teilnehmer sich im Netzbereich befinden und wie sie sich verhalten (das heißt, es ist kein Fehler, die »Null« eines Bereichs zu vergeben, in größeren heterogenen Netzen könnte es deswegen allerdings zu seltsamen Problemen kommen). Eine weitere besondere Adresse ist 127.0.0.1, die für den jeweiligen Rechner selber steht, sozusagen für Selbstgespräche. Zu IPv6 sei auf Kapitel 7.2.2 verwiesen.

Die Netzmaske definiert, wie oben bereits erwähnt, die Größe des lokalen Adreßbereichs; anhand der Zieladresse und der Netzmaske entscheidet der lokale Rechner, ob er ein Ziel direkt erreichen kann oder es nur über Vermittler (Router) erreichbar ist. Netzmasken sind Bit-Masken, eine nützliche Liste ist:

Netzmaske	CIDR-Notation	Anzahl Adressen	Anzahl Rechner
255.255.255.255	/32	1	1
255.255.255.254	/31	1	1
255.255.255.252	/30	4	2 (3)
255.255.255.250	/29	8	6 (7)
255.255.255.248	/28	16	14 (15)
255.255.255.240	/27	32	30 (31)
255.255.255.224	/26	64	62 (63)

```
255.255.255.128  /25          128          126 (127)
255.255.255.0    /24          256          254 (255)
```

Die CIDR-Notation (Classless Inter-Domain-Routing;) gibt an, wieviel Einsen in einer Netzmaske enthalten sind.

Der Router nach draußen

Eintragen

Normalerweise gibt es genau einen Weg aus dem lokalen Netz nach draußen. Dessen IP-Adresse (die im lokalen Netz) trägt man in die »/etc/rc.conf« als Default-Router ein, zum Beispiel mit

```
defaultrouter=192.168.1.1
```

Man kann die Defaultroute auch stattdessen in eine Datei »/etc/mygate« schreiben, sie in die »/etc/rc.conf« einzutragen ist aber aufgeräumter.

Prinzipiell werden alle Routen mit dem *route*-Befehl gesetzt[1]. Wenn man mehr Routen als die Defaultroute braucht, trägt man die weiteren als ausgeschriebenen Befehl in »/etc/netstart.local« ein. Wenn sie an ein Bestimmtes Interface gebunden sind, schreibt man sie in die »/etc/ifconfig.<if>«-Datei mit einem »!« (Ausrufezeichen) am Anfang der Zeile.

Wozu dient das?

In der Erläuterung zu IP-Adressen hatte ich erwähnt, daß für eine erfolgreiche Kommunikation Zieladresse und Absendeadresse eingetragen sein müssen. Wenn die Zieladresse nicht direkt erreichbar ist, braucht der lokale Rechner aber auch noch die Information, an wen er nicht-lokale Nachrichten zur Weiterleitung schicken kann. Dieses »Router« genannte Postamt muß dann den nächsten Zuständigen kennen und so weiter, wobei die Router auf dem Weg zunehmend nicht nur ihre lokale Gemeinde kennen werden, sondern auch die Wege in die Nachbarschaft. Die IP-Adresse enthält als klassische Adresse routing-relevante Informationen, wobei aber der spezifische Adressat nicht wie bei Postadressen am Anfang, sondern am Ende der Adresse steht. So befinden sich beispielsweise Adressen, die mit 194 anfangen, so gut wie immer in Europa (oder Nordafrika), und der einzelne Rechner wird bei einer Adresse A.B.C.D (A bis D seien ganze beliebige Zahlen zwischen 0 und 255) durch den hinteren Teil bezeichnet, bei dem die Netzmaske Null ist (also beispielsweise bei 255.255.255.0 ganz D).

---

[1] Von dynamischem Routing abgesehen, das nicht in den Rahmen dieses Buchs paßt. *routed* ist auf dem System vorhanden. Dynamisches Routing ist in den meisten LANs absolut überflüssig und man sollte sich an statischen Routen festhalten, solange dynamisches Routing nicht eine deutliche Vereinfachung verspricht.

Prinzipiell kann A.B.C dabei einen weltweit bekannten, herausgehobenen Weg bezeichnen, üblicher ist es allerdings, daß für andere Provider der Weg in A.B (und unter Umständen den ersten vier Bit von C) steckt und erst der eigene Provider wissen muß, was er mit dem Rest der Adresse anfangen soll. Aus diesem Grund ist es auch hoffnungslos, sich von einem Bekannten in einem anderen Netz eine freie weltweit eindeutige Adresse zu leihen – die Antworten werden nicht zurückfinden, sondern beim Bekannten landen.

Konfigurationen zur Abfrage des Domain Name Service

Konfiguration eines Nameserver-Clients

Die Adressen der Nameserver erfährt man entweder von seinem Netzwerkadministrator oder von seinem Provider. Eingetragen werden die IP-Adressen des/der Nameserver in »/etc/resolv.conf« (mit A bis H ganzen Zahlen zwischen 0 bis 255, die die Adresse bilden):

```
nameserver A.B.C.D
nameserver E.F.G.H
```

Konfiguration eines resolving Nameservers

Eintragen des Startbefehls (beim Booten) für den Nameserver in »/etc/rc.conf«:

```
named=YES
```

Das ist alles wirklich notwendige, der Default konfiguriert bereits einen lauffähigen resolving Nameserver. Jedoch sollte man noch folgende Schritte für die Sicherheit unternehmen:

- Starten in der (vorbereiteten) chroot-Umgebung:

  In »/etc/rc.conf« *named_chrootdir="/var/chroot/named"* eintragen und bequemlichkeitshalber einen Link von »/etc« dorthin setzen:

  ```
  ln -s /var/named/etc/namedb /etc/namedb
  ```

- Begrenzen der Rechner, die den Nameserver benutzen dürfen:

  In »/etc/namedb/named.conf« den lokalen IP-Adreßbereich (in der Form Netzbereich-Basisadresse/CIDR-Netzmaske) eintragen:

  ```
  acl localclients {
    lo.ca.ln.et/CIDRmask;
  };
  options {
    allow-query         { localclients; };
  };
  ```

Ein Forwarder wird absichtlich nicht eingetragen, da Forwarder nur dann sinnvoll sind, wenn der lokale Nameserver nicht wirklich eine Verbindung ins Internet hat, sondern einen Nameserver auf beziehungsweise auf der anderen Seite einer Firewall konsultieren muß. Ein Nameserver braucht nicht einen Forwarder wie ein Client seinen resolving Nameserver, sondern die Root Zone, die in jeder Nameserverinstallation enthalten[1] ist.

Konfiguration autoritativer Nameserver

Grundlagen zum »Distributed Name Service« DNS

Namensauflösung ist das Telefonbuch des Internets. So wie Telefonie auf Rufnummern (von Anrufer und Angerufenem) zur Auffindung von Kommunikations-Anschlüssen beruht, beruht die Identifikation von Absendern und Zielen von Kommunikation mit TCP/IP auf IP-Adressen. Da es aber eher mühsam ist, sich beispielsweise *198.41.0.4* zu merken statt *a.root-servers.net*, gibt es eine automatische Umsetzung von Namen zu Adressen und Adressen zu Namen.

Die nötigen Abfragen erledigt üblicherweise nicht jedes Programm für sich, sondern es gibt einen »Resolver« genannten Mechanismus, der anhand einer Konfigurationsdatei »resolv.conf« Datenquellen abklappert, bis er eine Antwort gefunden hat oder die Suche ergebnislos aufgibt. Dem Resolver stehen verschiedene Datenquellen zur Verfügung:

- */etc/hosts:* Moderne Mobiltelefone besitzen üblicherweise ein Telefonbuch, in dem aber nur die Telefonnummern stehen, die man selbst eingetragen hat. Die Funktion eines im Unfang begrenzten, lokalen Telefonbuchs übernimmt unter Unix die Datei »/etc/hosts«. Es ist im allgemeinen nützlich, sicherheitsrelevante Namens-/Adreßkombinationen sowie unter Umständen die Adresse des oder der Nameserver, die man verwendet, dort einzutragen; oft sind Netzwerkprobleme in Wahrheit Probleme mit der Namensauflösung und ein *ping* oder *traceroute* auf eine IP-Adresse oder einen in der hosts-Datei eingetragenen Namen bringt einem die Natur des Problems schnell näher.
- NIS (siehe Kapitel 7.6.5).
- Das Domain Name System: DNS ist ein hierarchisches System, bei dem üblicherweise keine Ebene »alles« weiß, sondern jede Ebene die »eigenen« Informationen hält sowie die Informationen, wer genauere Informationen hat. DNS kann bis zu 64 Ebenen tief geschachtelt sein, heutzutage ist eine tiefere Schachtelung als bis zur dritten Ebene selten geworden. So sollte beispielsweise auch die Polizei in Baden-Württemberg nicht »polizei-bw.de« verwenden, sondern »polizei.baden-wuerttemberg.de«, wie die anderen Landespolizeien es mit ihren Landes-Domains

[1] Der nicht seltene Fehler, zwei Nameserver sich gegenseitig als Forwarder einzutragen, führt nicht etwa zu Ausfallsicherheit, sondern zu broadcaststurm-ähnlichen Netzwerkzusammenbrüchen infolge einer endlosen Schleife für jede Nameserverabfrage. Außerdem funktioniert dann regelmäßig Namensauflösung nur extrem langsam bis gar nicht.

tun. (Allerdings kann dies bei diesem Bundesland-Namen verstehen, denn das ergäbe ziemlich lange Aufkleber.)

DNS ist ein »informatischer« Baum, der sich dadurch auszeichnet, daß die Wurzel oben liegt: Derzeit versorgen dreizehn Nameserver mit identischen Informationsgehalten das Internet mit der Root-Zone, der Zone (so nennt man eine geschlossene Datengruppierung im DNS) mit dem schönen vollständigen Namen ».« (in Worten: Punkt).

Diese Root-Zone enthält Informationen, wer auf der nächsten Ebene, der »Top Level Domain«, abgekürzt TLD, Informationen vorhält. Beispiele von TLDs sind zum Beispiel die Country Code TLDs (ccTLDs) *at.* (Österreich), *ch.* (Schweiz) und *de.* (Deutschland), oder die generic TLDs (gTLDs) *com.*, *org.*, *net.*, *gov.*, *int.*, *info.* und so weiter.

Ein Verweis von einer Hierarchie-Ebene auf die nächste heißt Delegation. Eine Delegation kann immer nur von »oben« (weniger spezifisch) nach »unten« (spezifischer) stattfinden. Beispielsweise ist in der nächsten Ebene unter *de.* die Second Level Domain (SLD) *netbsd.de.* (die hier als beliebiges Beispiel verwendet wird, aber auch tatsächlich existiert) auf die Nameserver *duck.ibh.net*, *micky.ibh.net* und *adolar.ibh.net* delegiert. Diese halten die Zone *netbsd.de*, in der zum Beispiel die Information »www.netbsd.de hat zur Zeit die Adresse 62.208.181.200« enthalten ist. In dieser Zone steht ebenfalls die Information, wer denn die Nameserver für *netbsd.de* sind, aber eine Änderung der Nameserver für *netbsd.de* in der Zone *netbsd.de* hat keine Auswirkung auf die Delegation an sich, und wenn man beispielsweise einen vierten Nameserver für die Domain dort eintragen würde, würde dieser nie nach den Zonendaten gefragt (es sei denn, dieser wäre ein Nameserver der Ebene darüber, und dann hätte man den subtilen Fehler namens »Zone Poisoning« vor sich). Warum dem so ist, ist ganz einfach zu verstehen, wenn man dem Resolver bei der Arbeit zusieht: Resolvt werden soll *www.netbsd.de.* Der Resolver fragt seinen Nameserver, und wenn der nicht zufällig selbst für die Daten zuständig ist, fragt dieser weiter:

Resolver Client	⇨	Resolving Nameserver: wer ist denn *www.netbsd.de.*?
Resolving Nameserver	⇨	Root Nameserver: Die Adresse zu *www.netbsd.de.*?
Root Nameserver	⇨	Resolving Nameserver: Zuständig für *de.* ist *<liste>*.
Resolving Nameserver	⇨	*de.* Nameserver: Die Adresse zu *www.netbsd.de.*?
de. Nameserver	⇨	Resolving Nameserver: Zuständig für *netbsd.de.* ist *<liste>*.
Resolving Nameserver	⇨	*netbsd.de.* Nameserver: Die Adresse zu *www.netbsd.de.*?
netbsd.de. Nameserver	⇨	Resolving Nameserver: *www.netbsd.de.* ist 62.208.181.200.
Resolving Nameserver	⇨	Resolver Client: *www.netbsd.de.* ist 62.208.181.200.

Aus dieser Abfolge wird klar, daß es völlig egal ist, was ein »nicht delegierter Nameserver« bezüglich der Zone *netbsd.de* glaubt zu wissen; normalerweise kann er nur seine eigenen Resolver-Clients verwirren, der Rest des Internets bekommt seine Meinungen gar nicht erst mit. Dies wird hier so ausführlich erläutert, weil es *die* Möglichkeit zur Fehlkonfiguration des DNS ist: Ein Verwalter einer Zone hat die Nameserver der Zone »geändert«, ohne jemals die Delegation ändern zu lassen, für ihn selber funktioniert ja alles, da er immer nur seine eigenen Nameserver fragt, und der Rest der Welt erreicht die Domain nur noch sporadisch.

Um von einer Adresse zu einem Namen zu kommen (genannt »Rückwärtsauflösung« oder Reverse Resolving) ist das Verfahren ähnlich, aber es wird zum Beispiel für die Adresse 1.2.3.4 nicht der root-Nameserver nach »4.« gefragt, sondern die Adresse wird gedreht, um die bei Domains übliche Reihenfolge zu erreichen (das spezifische steht vorne, das allgemeine hinten), und die symbolische Domain *in-addr.arpa* für IPv4-Adressen und *ip6.arpa* für IPv6-Adressen angehängt, so daß statt 1.2.3.4. nach *4.3.2.1.in-addr.arpa.* gefragt wird.

Domainnamen und Reverse sind vollkommen disjunkt, und daß beispielsweise *www.netbsd.de* zur Zeit die Adresse 62.208.181.200 trägt, 62.208.181.200 aber nicht das reverse *www.netbsd.de*, ist völlig normal. Wenn *forward* und *reverse* übereinstimmen, liegt das nur daran, daß die jeweiligen Verantwortlichen für die betreffenden Zonen das »zufällig« so eingetragen haben. Das ist der zweite übliche Fehler im DNS: Zu erwarten, daß sich ein *reverse* ändert, weil man das *forward* geändert hat.

Konfiguration einer DNS-Datenquelle (Master, Primary)

Anlegen der Zonen-Dateien:

Gegeben sei eine Domain *hier.invalid* sowie ein Adreßbereich 10.1.1.0/24.

Man konfiguriere zuerst einen Resolving Nameserver wie bereits angegeben. Dann lege man in »/etc/namedb« zwei Dateien an, beispielsweise *hier.invalid* und *10.1.1* genannt, sie sollen die Zonen der Domain und des Reverse-DNS des Adreßbereichs enthalten. Eine Zone besteht aus einem SOA- (Start of Authority Record), NS- (Name Server) Eintrag oder Einträge, und einer Default-TTL (Time To Live, Gültigkeitsdauer):

```
; Kommentare fangen mit ';' an
; Zeilen, die mit Tab beginnen, sind Fortsetzungszeilen
; Elemente eines Records werden mit Tab getrennt
; Tabs werden hier zur Verdeutlichung durch <tab> ersetzt
; In Ihrer Zonendatei muß <tab> natürlich wieder
; durch einen Tabulator ersetzt werden.
; Namen, die nicht mit einem '.' enden, bekommen den Domainnamen angehängt
;
; Gültigkeitszeit einer Auskunft: eine Stunde
$TTL 3600
; der SOA-Record
@               ; Variable für den Namen der geladenen Zone
```

```
<tab>IN          ; es handelt sich um einen IP-Record
<tab>SOA         ; es ist ein SOA-Record
<tab>nameserver1.hier.invalid. hostmaster.hier.invalid. ( ; wer ist
                                                          ; autoritativ?
; Name des Nameservers mit den Quelldaten, der 'Master' oder 'Primary'
; Mail-Adresse des Zonen-Verwalters mit '.' statt '@'
<tab>2003010100   ; serial = Versionsnummer der Zone, typischerweise Datum
; Die Software interpretiert die serial aber als
; ganze Zahl - mischen Sie deshalb NIE 8-stelliges Datum
; mit 6-stelligem Datum! Die Zahl darf - und muß - immer
; nur größer werden!
<tab>10000    ; refresh, wie oft Secondaries die serial kontrollieren$
<tab>7200     ; retry, wie schnell ein fehlgeschlagener Refresh wiederholt
              ; wird$
<tab>604800   ; expire, wie lange ein Secondary die Zone behält
              ; wenn der Primary nicht erreichbar ist$
<tab>600 )    ; minimum, wie lange eine "gibt es nicht"-Auskunft gültig
              ; ist$
; Nameserver-Record
                ; nichts in Position 1: zweiter Record zu '@'
<tab>IN          ; es handelt sich um einen IP-Record
<tab>NS          ; es ist ein NS-Record
<tab>nameserver1 ; Name eines delegierten Nameservers
; Ende des Pflichtteils
```

Die übliche Schreibweise ist:

```
$TTL 3600
@       IN      SOA   nameserver1.hier.invalid. hostmaster.hier.invalid. (
                      2003010100    ; serial
                      10000         ; refresh
                      7200          ; retry
                      604800        ; expire
                      600 )         ; minimum
        IN      NS    nameserver1
```

Sowie den Daten, für die man sie braucht, das heißt, PTR (PoinTeR) für Reverse Zonen und typischerweise A (Adreß), MX (MaileXchanger) und CNAME (Canonical Name, generischer Name) für Domain-Zonen:

```
; Benennung des Mailservers, der Mail für user@hier.invalid annimmt
                    ; nichts in Position 1: Fortsetzung zu '@'
<tab>IN             ; IP-Record
<tab>MX             ; Mailexchanger
<tab>10 mailserver  ; Metrik und Name des Mailservers
                    ; je kleiner die Metrik, desto bevorzugter ist der
```

```
                    ; Mailserver
; Ein Adreß-Record für den Nameserver1
nameserver1         ; Name
<tab>IN             ; IP-Record
<tab>A              ; Adreß-Record
<tab>10.1.1.23      ; IP-Adresse von nameserver1
; mail an nameserver1 geht an mailserver
                    ; nichts in Position 1: Fortsetzung zu nameserver1
<tab>IN             ; IP-Record
<tab>MX             ; Mailexchanger
<tab>10 mailserver  ; Metrik und Name des Mailservers
; Ein Adreß-Record für den Mailserver
mailserver          ; Name
<tab>IN             ; IP-Record
<tab>A              ; Adreß-Record
<tab>10.1.1.42      ; IP-Adresse von mailserver
; ein CNAME-Record, da Mailserver auch WWW-Server ist und als www
; erreichbar sein soll
www                 ; Name
<tab>IN             ; IP-Record
<tab>CNAME          ; CNAME-Record
<tab>mailserver     ; Name des Rechners, für den www ein Alias ist
; mail an www geht an mailserver
                    ; nichts in Position 1: Fortsetzung zu www
<tab>IN             ; IP-Record
<tab>MX             ; Mailexchanger
<tab>10 mailserver  ; Metrik und Name des Mailservers
```

Zusammen mit dem SOA in typischer Schreibweise:

```
$TTL 3600
@   IN  SOA nameserver1.hier.invalid. hostmaster.hier.invalid. (
            2003010100  ; serial
            10000       ; refresh
            7200        ; retry
            604800      ; expire
            600 )       ; minimum
    IN  NS  nameserver1
    IN  MX  10 mailserver
;
nameserver1     IN      A           10.1.1.23
                IN      MX          10 mailserver
mailserver      IN      A           10.1.1.42
www             IN      CNAME       mailserver
                IN      MX          10 mailserver
;
```

Bei der »Befüllung« einer Zone muß darauf geachtet werden, daß »rechts« nur Namen zu stehen kommen, die selbst einen A-Record besitzen, also kein CNAME sind. Damit soll verhindert werden, daß die Kette der Abfragen gar zu lange wird, weil beispielsweise *www* eine CNAME von *server* ist, der ein CNAME von *Server-in-Stadt-A* ist, der ein CNAME von *idefix* ist, dessen A-Record irgendwann abgeschafft wird, woraufhin sich alle wundern warum *www* nicht mehr auffindbar ist. Einträge »rechts« können, müssen aber nicht in der gleichen Zone liegen. Wenn sie aus einer anderen Zone kommen, *müssen* die Namen mit einem Punkt enden, damit der Zonenname nicht angehängt wird.

Um zu verhindern, daß man bei der Eintragung eines Namens in der Domain-Zone den Reverse-Eintrag vergißt, kann man die eigentlichen Pointer-Einträge auch in die Domain-Zone verlagern. Dazu trägt man in die Reverse-Zone keine Pointer ein, sondern CNAMEs, die in die betreffende Domain zeigen, also:

```
1       IN      CNAME       pointer1.hier.invalid.
2       IN      CNAME       pointer2.hier.invalid.
3       IN      CNAME       pointer3.hier.invalid.
4       IN      CNAME       pointer4.hier.invalid.
23      IN      CNAME       pointer23.hier.invalid.
42      IN      CNAME       pointer42.hier.invalid.
```

Woraufhin die Domain-Zone wie folgt ergänzt wird:

```
$TTL 3600
@   IN  SOA nameserver1.hier.invalid. hostmaster.hier.invalid. (
            2003010100  ; serial
            10000       ; refresh
            7200        ; retry
            604800      ; expire
            600 )       ; minimum
    IN  NS  nameserver1
    IN  MX  10 mailserver
;
nameserver1 IN  A       10.1.1.23
            IN  MX      10 mailserver
pointer23   IN  PTR     nameserver1
;
mailserver  IN  A       10.1.1.42
pointer42   IN  PTR     mailserver
www         IN  CNAME   mailserver
            IN  MX      10 mailserver
;
```

Diese Methode, Reverse-DNS in eine Domain-Zone zu verlagern, ist auch die offizielle Methode, Nutzern von weniger als /24 offiziellen Adressen am Stück direkte Kontrolle über ihr Reverse-DNS zu geben und trägt deswegen den Namen CIDR-Reverse. Wenn

der Nutzer keine »eine eindeutige« Domain hat, legt man eine Sub-Zone zur Reverse-Zone an, also beispielsweise *0-63.1.1.10.in-addr.arpa*, die man aus der *1.1.10.in-addr.arpa* herausdelegiert:

```
$TTL 3600
@   IN  SOA nameserver1.hier.invalid. hostmaster.hier.invalid. (
            2003010100  ; serial
            10000       ; refresh
            7200        ; retry
            604800      ; expire
            600 )       ; minimum
    IN  NS  nameserver1.hier.invalid.
;
; Delegation von 0-63 auf nameserver1.dort.invalid.
0-63    IN      NS              nameserver1.dort.invalid.
; Delegation von 0-63 auf nameserver1.hier.invalid.
0-63    IN      NS              nameserver1.hier.invalid.
;
1   IN  CNAME   pointer1.0-63.1.1.10.in-addr.arpa.
2   IN  CNAME   pointer2.0-63.1.1.10.in-addr.arpa.
3   IN  CNAME   pointer3.0-63.1.1.10.in-addr.arpa.
4   IN  CNAME   pointer4.0-63.1.1.10.in-addr.arpa.
23  IN  CNAME   pointer23.0-63.1.1.10.in-addr.arpa.
42  IN  CNAME   pointer42.0-63.1.1.10.in-addr.arpa.
;
```

In obigen Beispielen ist zum Teil nur ein Nameserver in den Zonen eingetragen. Das ist für eine Zone, die nur zuhause genutzt wird, durchaus tragbar, offizielle Zonen sollten aber mindestens zwei Nameserver enthalten, die möglichst nicht am gleichen Ort stehen, also vom Ausfall einer Leitung nicht beide lahmgelegt werden. Es gibt praktische Unterschiede zwischen »das Ziel ist nicht erreichbar« und »das Ziel ist nicht auffindbar«, die beispielsweise beim Mailversand unmittelbar zuschlagen.

Eintragen der Zonen in die Nameserver-Konfiguration

In »/etc/namedb/named.conf« werden jetzt noch die fertigen Zonendateien eingetragen:

```
zone "hier.invalid" {
     type master;
     file "hier.invalid";
};

zone "1.1.10.in-addr.arpa" {
     type master;
     file "10.1.1";
};
```

Und der Nameserver wird mittels *ndc reconfig* oder Neustart angewiesen, seine Konfigurationsdatei neu zu lesen.

Wenn es für die Zone Slaves (Secondaries, »Backup-DNS-Server«, hier als Beispiel mit der Adresse 10.2.2.42) gibt, sollte man ihnen auch noch den Zonentransfer erlauben:

```
zone "hier.invalid" {
   type master;
   file "hier.invalid";
   allow-transfer {10.2.2.42;};
};

zone "1.1.10.in-addr.arpa" {
   type master;
   file "10.1.1";
   allow-transfer {10.2.2.42;};
};
```

Änderungen in den Zonen im Betrieb

Wenn sich an den Informationen der Zonen, für die der Nameserver autoritativ ist, etwas ändert, trägt man die Änderung in der Zonen-Datei ein, setzt eine neue Versionsnummer und teilt nach dem Abspeichern dem Nameserver mittels *ndc reload <zone>* mit, daß er die Zone neu lesen und gegebenenfalls vorhandene Slave-Nameserver über die Änderung informieren soll.

Konfiguration eines Slave Nameservers

Um einen Slave (oder Secondary) für eine Zone aufzusetzen, trägt man folgendes in »/etc/namedb/named.conf« ein:

```
zone "dort.invalid" {
      type slave;
      masters 10.2.2.42;
};
```

und führt ein *ndc reconfig* aus. Das war's!

7.4 NetBSD als Firewall und Router

von Markus Illenseer

Ein Router ist ein Rechner, der normalerweise ausschließlich die Aufgabe hat, eine Verbindung zwischen zwei oder mehreren verschiedenen Netzwerken aufrecht zu erhalten. Oft wird dabei in den verschiedenen Netzen eine andere Hardware zur Kommunikation verwendet, etwa ISDN und Ethernet, so daß der Router auch eine Transformierung und gar Drosselfunktion übernimmt, da ISDN langsamer als Ethernet ist und andere Paketgrößen verwendet. Ein Router soll aber vor allem die Aufgabe haben, die Wegfindung für TCP/IP-Anwendungen zwischen den Netzen zu automatisieren. Wir schauen uns einen sehr häufig vorkommenden Beispielfall an und erläutern die Einrichtung.

7.4.1 NetBSD-Router mit zwei Netzwerkkarten

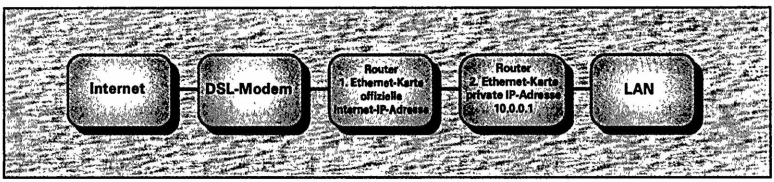
Sie haben ein lokales LAN. Ein paar Rechner sind mittels Ethernet zusammengeschlossen und können lokal Daten austauschen. In der Regel verwenden Sie dazu IP-Adressen aus dem zur privaten Nutzung freigegebenen Bereichen:

Privates Netzwerk	Subnetz-Maske	IP-Adreß-Bereich
10.0.0.0	255.0.0.0	10.0.0.1 - 10.255.255.254
172.16.0.0	255.240.0.0	172.16.0.1 - 172.31.255.254
192.168.0.0	255.255.0.0	192.168.0.1 - 192.168.255.254

Nur diese Adressen sind in einem privaten Netzwerk, das nicht direkt an das Internet ansgeschlossen wird, erlaubt. Sofern Sie lokal fiktive Adressen verwenden, also andere als oben angegeben, kann es Ihnen spätestens bei Anschluß an das Internet passieren, daß Sie statt des lokalen Nachbarrechners einen Rechner im Internet ansprechen, denn dieser Rechner im Internet besitzt die Adresse hochoffiziell.

Wir verwenden im Beispiel das lokale Netz 10.0.0.0 mit der Subnetz-Maske 255.255.255.0 – also alle Rechner zwischen 10.0.0.0 und 10.0.0.255. Die erste Adresse 10.0.0.0 ist die Netzwerkadresse. Diese können wir nicht für einen Rechner nehmen. Die erste freie Adresse ist damit 10.0.0.1. Diese teilt man in der Regel dem Router zu, der das Netz mit einem anderen Netz verbindet. Genauso kann man aber jede andere freie IP-Adresse dafür logisch definieren, ein Muß gibt es dabei nicht, denn man kann auch mehrere Router in einem Netz betreiben. Die IP-Adresse 10.0.0.254 ist die letzte IP-Adresse in diesem Netz, die für einen Rechner verwendet werden kann, da 10.0.0.255 die Broadcast-Adresse ist.

Unser unter NetBSD als Router festgelegter Rechner besitzt zwei Ethernet-Netzwerkkarten. Mit der einen Karte »redet« er mit dem lokalen LAN. Mit der anderen ist er beispielsweise mit einem DSL- oder ISDN-Modem beziehungsweise -Router verbunden. Die Einstellung für den Anschluß an das Internet hat uns der Internet Service Provider bereits gegeben, sie müssen für die Netzwerkkarte eingestellt werden, die mit dem Modem verbunden ist.



Unser Router besitzt nun die IP-Adresse 10.0.0.1 im lokalen LAN. Diese Adresse wird der ersten als lokal definierten Ethernet-Netzwerkkarte zugeweisen. Alle anderen Rechner in unserem LAN, die über diesen Router in das Internet gelangen sollen, bekommen die weiteren Adressen zugewiesen. Es ist wichtig zu wissen, daß man die IP-Adressen nur logisch im TCP/IP-Protokoll zuweist. Physikalisch, also im Ethernet, können noch viele weitere Rechner angeschlossen sein und völlig andere IP-Nummern erhalten. Auf TCP/IP-Ebene können sich Rechner nur dann unterhalten, wenn sie im selben IP-Subnetz definiert wurden.

Ein Rechner mit der IP-Adresse 192.168.0.2 kann sich nicht mit dem Rechner 10.0.0.4 unterhalten, auch wenn sie im selben physikalischen Netz und vielleicht dreißig Zentimeter entfernt voneinander stehen. Man definiert normalerweise eine IP-Nummer pro physikalischer Netzwerkeinheit, also pro Ethernet-Adapter. Aber gerade bei Servern und Routern kann man auch mehrere IP-Adressen pro Adapter eintragen, was die physikalischen Erfordernisse erleichtert (man braucht nur einen Adapter statt dreien).

Das klingt verwirrend, ist aber hochgradig flexibel – und auch gefährlich, da man Hakkern damit Tor und Tür öffnet. Es ist sinnvoll, pro physikalischem Netzwerk auch nur ein logisches IP-Netzwerk zu betreiben.

Unsere Rechner im LAN müssen nun noch wissen, was sie tun sollen, wenn eine Applikation eine IP-Adresse ansprechen möchte, die der Rechner nicht direkt selbst adressieren kann, weil er nicht in diesem Subnetz eingetragen ist. Dies nennt man »Default-Gateway« oder »Default-Route« und wird unter NetBSD in der Datei »/etc/rc.conf« eingetragen und beim Start des Rechners automatisch zugeweisen. Sollten die anderen Rechner im LAN nicht unter NetBSD laufen, muß das jeweilige Betriebssystem beziehungsweise dessen TCP/IP-Stack konfiguriert werden.

Der relevante Eintrag in der Datei auf einem Rechner im LAN sieht wie folgt aus:

```
hostname="albert.domain.de"
defaultroute="10.0.0.1"
```

Der relevante Eintrag in der Datei auf dem Router sieht wie folgt aus:

```
hostname="router.domain.de"
defaultroute="202.192.31.65"
```

Die IP-Adresse für die Default-Route hat Ihnen der Internet Service Provider genannt. Eventuell hat dieser Ihnen gesagt, daß das DSL-Modem alles weitere für Sie tut und eigentlich schon der Router ist. Leider hat so ein DSL-Modem nur wenige Konfigurati-

onsmöglichkeiten für eine Firewall und mehr. Deswegen bauen wir auch gerade einen Router unter NetBSD auf. In diesem Fall ist das DSL-Modem der Default-Router oder -Gateway. Tragen Sie also die IP-Adresse des DSL-Modems ein.

Der Eintrag und die Konfiguration kann nur als Benutzer *root* vorgenommen werden.

Der Rechner im LAN heißt also *albert* und befindet sich in der Domain *domain.de*. Die eigene IP-Nummer steht unter NetBSD in der Datei »/etc/ifconfig.netzkartenabkürzung«. Also etwa »/etc/ifconfig.epic0«. Diese Datei wurde bei der Installation von NetBSD angelegt. Mit dem Befehl *ifconfig -a* bekommen Sie heraus, welche Netzwerkkarten bei Ihnen vom System erkannt und initialisiert wurden.

Unsere Default-Route ist damit eingetragen. Weil wir jetzt nicht neu booten wollen, nur weil wir einen Default-Router eingestellt haben, starten wir das Skript */etc/rc.d/network* mit der Option *restart*:

```
%> /etc/rc.d/network restart
```

Vorsicht, wenn Sie diesen Befehl verwenden, wenn Sie die Konfiguration mittels Remote-Zugang per *telnet* oder *ssh* vornehmen – Sie sind dann bei einem Fehler sofort ausgeschlossen!

Die Überprüfung, ob die Einstellung funktioniert hat, wird mit Hilfe des Programms *route* und der Option *show* vorgenommen:

```
%> route show
Routing tables

Internet:
Destination       Gateway           Flags
default           10.0.0.1          UG
loopback          127.0.0.1         UG
localhost         127.0.0.1         UH
10.0.0.0          link#1            U
```

Wie Sie sehen, ist das Default-Gateway hier korrekt auf 10.0.0.1 eingetragen und weiter unten auch als über Ethernet *link#1* erreichbar markiert. Bitte konsultieren Sie die Manpage von *route*, um mehr über die Ausgabe zu erfahren.

7.4.2 Installation als Application-Gateway

Doch halt – wir haben etwas wesentliches übersehen. Das Szenario sah ja vor, ein privates Netzwerk an das Internet anzuschließen. Die einzigen Rechner, die bislang offiziell an das Internet angeschlossen sind, sind das DSL-Modem und der Router mit einer seiner Netzwerkkarten. Auf den Rechnern im lokalen LAN ist zwar eine Default-Route zum Router gelegt. Dieser leitet die Pakete dann auch brav zum Internet (also zum DSL-Modem), aber niemand im Internet weiß (beziehungsweise darf wissen) wie man unsere privaten Rechner zurück erreicht. Für die IP-Adressen, die als privat gekenn-

zeichnet sind, gibt es weltweit keine Rückroute. Jeder darf sie ja verwenden! Was tun? Wie bekommt man von einem lokalen Rechner aus Zugriff auf's Internet?

Hier kommen wir zum nächsten Szenario, das dringend von Profis auf diesem Gebiet bei Ihnen eingeführt werden sollte, falls Sie einen direkten Zugang aller Ihrer Bürorechner in das Internet planen. Dieses Szenario sieht eine Firewall vor, die unsere lokalen Rechner vor dem Unbill des Internet schützen soll. Vorerst sind wir sicher, daß nicht viel passieren kann. Unsere lokalen Rechner sind ja wegen des Tricks von oben – der Verwendung privater Adressen – aus dem Internet heraus gar nicht erreichbar. Es sei denn, jemand hackt unseren Router. Wie das zu vermeiden ist, wird gleich erklärt.

Wenn Sie mit den lokalen Rechnern nur mal ab und zu surfen, vielleicht noch E-Mail versenden wollen, genügt das Szenario aus TCP/IP-Sicht vorerst genügend. Unser Router kann nämlich problemlos als sogenanntes Application-Gateway eingerichtet werden. Das sind Applikationen, die das jeweilige Dienstprotokoll verstehen und Anfragen von intern nach extern durchreichen. Bekanntestes Beispiel ist der HTTP-Proxy-Cache *squid*, zu finden als NetBSD-Package unter */usr/pkgsrc/www/squid*. Dieser führt WWW-Anfragen durch und speichert (cacht) zudem noch die abgerufenen Seiten (und spart damit viel Geld, weil Objekte nicht mehrfach geladen werden müssen). Der lokale Rechner beziehungsweise der dort verwendete Web-Browser erhält dann als Proxy-Server unseren Router mit installiertem *squid* zugeweisen und schon kann er im Internet surfen, ohne selbst direkt im Internet erreichbar zu sein.

Auch kann der Router als Mail-Server fungieren, sowohl für den Versand mittels Sendmail als auch für den Empfang und das Einlagern der Mail und den Abruf der Mailbox per POP3 oder besser IMAP4 (für POP3 und IMAP4 gibt es NetBSD-Packages). Die lokalen Rechner laden die Mail dann vom Router und senden via Router ins Internet. Ein durchaus gängiges Konzept. Nur daß eigentlich ein Router nur routen sollte, mehr nicht. Lassen Sie sich also unbedingt beraten, wie in Ihrer Umgebung ein idealer Zugang zum Internet aussieht. Für den privaten Gebrauch ist die Installation von Diensten auf dem Router in Ordnung. Für den Gebrauch in einer Firma sollte davon abgesehen werden.

7.4.3 IP-NAT installieren

Nun kann es sein, daß es für einige Programme wie IRC oder *ftp* keine Software gibt, um als Application-Gateway zu fungieren. Dann benötigen wir eine andere Lösung, um unsere unter privater IP-Adresse laufenden Rechner im Internet agieren zu lassen. Dafür wurde das sogenannte »IP-NAT«-Verfahren entwickelt. NAT steht für »Network Address Translation« und wandelt die lokale Rechner-IP-Adresse in eine offziell im Internet nutzbare und zugeteilte IP-Adresse um. Das heißt, wir installieren auf unserem NetBSD-Router eine Software, um das IP-NAT-Verfahren einzusetzen. Alle Anfragen unserer lokalen Rechner werden nun von der Software auf die – in diesem Beispiel – einzige gültige Adresse des Routers umgesetzt, die uns unser Internet Sevice Provider gegeben hat. Die Software merkt sich in Tabellen, welcher lokale Rechner mit welchem Dienstprotokoll welchen Dienst im Internet auf welchem Server angefragt hat und leitet die erhaltenen Antworten zurück an den lokalen Rechner. Dies bedeutet aber immer noch, daß man keine Verbindung von extern aus dem Internet an unseren loka-

len Rechner senden kann, da man dort erstens nur die IP-Adresse unseres Routers eingeben kann, und dieser aber zweitens in der IP-NAT-Software nur ausgehende Verbindungen merkt und es ja bei eingehenden Anfragen keinen vorliegenden Eintrag in der Tabelle gibt. Es sei die Anmerkung erlaubt, daß man mit ein paar Tricks dies dennoch erlauben kann, dies aber in einem Firewall-Konstrukt nicht gestattet werden sollte.

IP-NAT zu konfigurieren ist nicht sehr schwer. Stellen Sie sicher, daß der NetBSD-Router die Option *ipfilter* im Kernel konfiguriert hat. Folgende Optionen müssen in der Konfigurationsdatei zum Kompilieren des Kernel existieren:

```
options         PFIL_HOOKS
packet filter hooks
pseudo-device   ipfilter                # IP filter (firewall) and NAT
```

Gegebenenfalls müssen Sie einen neuen Kernel kompilieren und damit starten.

Tragen Sie nun folgendes in die Datei »/etc/ipf.conf« ein:

```
pass in from any to any
pass out from any to any
```

(Eingehende und ausgehende Verbindungen zu unseren Rechnern zulassen.)

Die nächsten Zeilen tragen Sie in die Datei »/etc/ipnat.conf« ein:

```
map ppp0 10.0.0.0/24 -> 0/32 proxy port ftp ftp/tcp
map ppp0 10.0.0.0/24 -> 0/32 portmap tcp/udp 40000:60000
map ppp0 10.0.0.0/24 -> 0/32
```

Es wird explizit das Dienstprotokoll *ftp* erlaubt, aber auch alle TCP/UDP-Ports von 40000 bis 60000 sind erlaubt. Per Vorgabe sind außerdem Verbindungen zu allen TCP/IP-Ports erlaubt. Wohlgemerkt nur von innen aus dem LAN nach außen zum Internet.

Wir verwenden hier das lokale IP-Netzwerk 10.0.0.0. Sollten Sie andere Netzwerke einsetzen, müssen Sie die Einträge entsprechend anpassen. Die /24 ist eine andere Notation für die Angabe des Subnetzes. Das Subnetz beziehungsweise seine Größe wird über eine Netzmaske definiert. Sie erinnern sich noch? IP-Adressen sind 32 Bit lange Dual-Zahlen. Bei /24 ist gemeint, daß die 24 ersten Bits der 32-Bit-Zahl auf 1 gesetzt sind, die restlichen acht sind 0. Sprich: 255.255.255.0 in dezimaler Schreibweise. Bei /26 adressieren wir demnach ein 64-er Subnetz.

Jetzt müssen wir das Modul *ipfilter* im Kernel einschalten und bei Systemstart automatisch aktivieren. Tragen Sie folgendes in der Datei »/etc/rc.conf« ein:

```
ipfilter=YES        # benutzt Konfiguration aus /etc/ipf.conf
ipnat=YES           # benutzt Konfiguration aus /etc/ipnat.conf
```

Jetzt müssen wir dem Router noch mitteilen, daß er wirklich IP-Pakete von einem Netzwerk zu einem anderen Netzwerk weiterleiteten soll. Entweder ist dies im Kernel konfiguriert oder wir tragen es mit folgendem Befehl direkt im Kernel ein:

```
%> sysctl -w net.inet.ip.forwarding=1
```

Diese Kernel-Einstellung während des laufenden Betriebs ist ein Eingriff in die Sicherheit Ihres Rechners, da ab sofort mehrere Netzwerke des Routers gleichzeitig und unter Umständen auch vom Internet aus direkt erreichbar sind! Ohne diese Einstellung werden keine IP-Pakete weitergereicht. Wir brechen damit also unsere »Isolation« auf und werden Teil des großen Internet – mit all seinen Folgen. Die Einstellung kann nur als User *root* vorgenommen werden.

Zur Überprüfung, ob wir alle Schritte korrekt durchgeführt haben, sollten Sie jetzt den Router booten.

Jetzt haben wir einen Router, der als Application-Gateway funktionieren kann und außerdem noch unsere lokalen Rechner mit Hilfe von IP-NAT nahezu frei im Internet beweglich macht.

Ein anderes denkbares Szenario ist, daß am Router ein Modem an der seriellen Schnittstelle angeschlossen wird, das dann mit Hilfe des logischen Netzwerkinterfaces *ppp* die Verbindung zum Internet aufbaut. Mehr zu diesem Szenario gibt es auch in den Netzwerk-Dokumentationen auf dem NetBSD-Webserver.

Die folgenden Befehle dienen der Administration und Überwachung des Netzwerks und seiner Elemente unter NetBSD:

ifconfig(8)	Konfiguriert Netzwerk-Adapter.
route(8)	Manipuliert die Wegfindung.
ping(8)	Sendet einen ICMP ECHO_REQUEST an einen oder mehrere Rechner (in Anlehnung an das gute alte Echolot bei U-Booten).
traceroute(8)	Listet die Route auf, die Pakete vom lokalen Rechner zu einem Zielrechner nehmen.
tcpdump(1)	Zeigt ausgewählten Verkehr auf angegebenen Netzwerk-Adaptern.
netstat(1)	Zeigt verschiedene Netzwerk-Stati an.
systat(8)	*systat netstat* zeigt Netzwerkverbindungen dynamisch an.

Zu jedem Befehl gibt es eine ausführliche Manpage. Bei jedem dieser Befehle gilt: Lesen – Informationen aus dem Netzwerk – darf nahezu jeder Systembenutzer. Schreiben – beispielsweise die Netzwerkadapter konfigurieren – darf nur *root*. Die Netzwerkeinstellungen werden in verschiedenen Konfigurationsdateien definiert:

Datei	Funktion
/etc/rc.conf	*rc.conf*(5) spezifiziert Systemdienste und Netzwerkdienste, die beim Systemstart automatisch gestartet werden sollen.
/etc/hosts	*hosts*(5) definiert essentielle Zuweisungen von Rechnernamen zu IP-Adressen.
/etc/myname	Der eigene voll qualifizierte Rechnername, mit Angabe von Domain, falls der Rechner in ein DNS-System integriert ist.
/etc/mygate	Das Default-Gateway. Eigentlich als Eintrag in der Datei »/etc/rc.conf« besser aufgehoben; alternativ kann auch der *routed*(8) gestartet werden.
/etc/ifconfig.{IF}.	Die Definition des Netzwerk-Hardwareinterfaces *IF*, das durch das Skript */etc/rc.d/network* bei Systemstart ausgelesen und verwendet wird, um ein Netzwerkinterface zu setzen.
/etc/ifaliases	*ifaliases(5)* weist einem Netzwerkinterface mehr als eine IP-Adresse zu. Aliase werden hier spezifiziert und durch »/etc/rc.d/network« initialisiert.

Tabelle 7.4: Konfigurationsdateien für Netzwerke

7.5 Sicherheit mit ssh und ssl

von Markus Illenseer

Die Sicherheit des Systems ist immer wichtig, egal, ob der Server in einem lokalen LAN oder gar im Internet agiert. Aus den Urzeiten der unix-ähnlichen Betriebssysteme gibt es immerhin verschlüsselte Paßwörter der System-Benutzer. Auch heute ist dies bei NetBSD immer noch Standard. Was sich aber seitdem geändert hat, ist die Art und Weise, wie man seine Daten über ein Netzwerk hin zum Server reicht, um sich etwa dort einzuloggen oder Daten zu übertragen. Nicht nur, daß man Account-Namen und -Passwort verschlüsselt übertragen sollte, sondern auch die danach versandten Daten sollten immer verschlüsselt werden. Wenn man sich so etwas angewöhnt, ist man zumindest vor einfachen Lücken gefeit, die Netzwerkdienste mit sich bringen, deren Account-Daten im Klartext verschlüsselt werden.

Die folgenden Sicherheitsmechanismen sollen hier erklärt werden:

- *ssh* und *sshd* als Ersatz für die sogenannten *r*-Tools und *telnet*
- *ssl* als Zusatz zum Apache-Webserver

7.5.1 ssh – Die Secure Shell

Die Programmsammlung um *ssh* gehört zum Basissystem von NetBSD und ist eines der beliebtesten Tools im System beim Anschluß an ein Netz.

ssh ist abgeleitet aus »Secure Shell« und umfaßt einige Tools, die man hervorragend zur Fernwartung oder -arbeit verwenden kann. Im Grunde sind sie eine Variante der sogenannten »r-Tools«, bei der alle Daten verschlüsselt über das Netz gehen. Es ist nicht mehr angeraten, auch nicht aus Bequemlichkeit, die alten Tools rund um *rsh* und *rlogin* oder – noch älter – *telnet* überhaupt noch am Leben zu erhalten. Jedes Skript oder Programm, das mit *rsh* funktioniert, kann an *ssh* angepaßt werden.

Die Gruppe der Tools aus der ssh-Serie umfaßt vor allem folgende Programme:

- *ssh* – Secure Shell für Fernarbeiten auf einem anderen Rechner.
- *scp* – Secure Copy für das Kopieren von Daten von einem Rechner zu einem anderen.
- *sftp* – Secure *ftp* – Verschlüsseltes *ftp*.

ssh gibt es als Client für einen Arbeitsplatz und als Daemon (*sshd*) für den entfernten Server. *ssh* benötigt also einen laufenden Daemon auf der anderen Maschine, mit der man arbeiten möchte, damit es dort verbinden und Daten austauschen kann. Der Daemon wird in der Konfigurationsdatei »/etc/rc.conf« mit dem Eintrag *sshd=YES* bei Systemstart automatisch gestartet. Dies ist wichtig, damit man sich auch wirklich zum Server verbinden kann. Die Skript-Datei »/etc/rc.d/sshd« startet, stoppt oder prüft – wie gewohnt von rc-Skripten – den Daemon.

Um nun auf den Server zuzugreifen, verwendet man auf einem Arbeitsplatz das Programm *ssh*:

```
%> ssh anton@server
```

Nun werden Sie gefragt, ob Sie wirklich zu diesem bislang nicht bekannten Server Verbindung aufnehmen wollen. Die Sitzung sieht etwa wie folgt aus:

```
anton@arbeitsplatz01:~>ssh anton@server
The authenticity of host 'server (192.168.0.2)' can't be established.
DSA key fingerprint is 07:c8:cc:2b:7d:02:57:8d:00:45:c9:2f:f9:b7:e4:9c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server,192.168.0.2' (DSA) to the list of known
hosts.
anton@server's password:
Last login: Mon May 27 14:43:58 2002 from 195.143.198.40
Have a lot of fun...
```

Übersetzt bedeutet dies, daß die Authentizität des Servers nicht festgestellt werden konnte und *ssh* daher nicht verbinden konnte. Der Benutzer wird gefragt, ob man sicher sei und sich wirklich zu diesem Server verbinden möchte, was dieser bejaht. Danach wird der Verschlüsselungsprozeß für die Übertragung des Passworts eingeleitet. Der Benutzer authentifiziert sich nun gegenüber dem regulären System und logt sich als Benutzer *anton* ein.

Im lokalen Home-Verzeichnis des Benutzers *anton* auf dem *arbeitsplatz01* wird das Verzeichnis ».ssh« angelegt, in dem nun eine Datei »known_hosts« vorhanden ist. Dort wird der *server* mitsamt einer eindeutigen ID abgelegt. Ändert sich nun auf dem Server zum Beispiel die IP-Adresse oder die ID (aus verschiedenen Gründen), dann bemerkt *ssh* das und warnt den Benutzer. Es könnte ja immerhin sein, daß jemand den Server gehackt hat!

ssh erlaubt auch das Ausführen von Programmen auf dem Server, ohne sich dort mit einer Shell einzuloggen:

```
%> ssh anton@server ls -al
anton@server's password:
insgesamt 1
drwxr-xr-x    3 anton   users        4096 Apr 12 16:31 data
```

Es wurde ein *ls -al* auf dem Server »server« mit der Nutzerkennung *anton* gestartet. ssh verhält sich also so, als sei man physikalisch am Server angemeldet und würde dort ein *ls -al* starten. Die Ausgabe wird auf das lokale Terminal umgelenkt.

Mit dem Tool *scp* kann man nun auch Dateien auf sichere Art und Weise vom Arbeitsplatz zum Server übertragen:

```
%> scp anton@server:ausschreibung.txt ausschreibung27.txt
```

Dies kopiert die Datei »ausschreibung.txt», die sich beim User *anton* auf dem Server *server* befindet, in das lokale Verzeichnis (mit anderem Namen). Auch hier werden – bei Erstverwendung des Servers – das Verzeichnis ».ssh« und die Konfigurationsdateien dort berücksichtigt und Sicherheitslücken erkannt. Wichtig zu wissen nun, daß die Ausschreibung nun nicht im Klartext über das Netz geht und von nahezu jedermann gelesen werden kann.

7.5.2 Apache-Server mit ssl installieren

Oft muß und will man seinen Benutzern im Internet oder Extranet eine geschützte und zertifizierte Verbindung zur Übertragung sensibler Daten zur Verfügung stellen. Als Benutzer erkennt man das, wenn der geladene URL mit *https* anfängt. Das »s« steht dann für »secure«, also »sichere Verbindung«. Der Apache benutzt dafür das Modul *mod_ssl*.

Eigentlich ist es an sich ein sehr kompliziertes Unterfangen, *ssl* zu konfigurieren und das benötigte Zertifikat einzuspielen. Die Anleitungen von *mod_ssl* und auch OpenSSL als Basispaket sind etwas ungenau. Aktuelle Veröffentlichungen tun sich schwer, das Verfahren klar darzustellen. Auch wir müssen bei der Komplexität des Themas Abstriche machen und beschränken uns auf die Bereiche des Erstellens eines privaten Zertifikats und das Einrichten des Apache mit *ssl*.

Die https-Verbindung, die verschlüsselte Verbindung von http, des Web-Protokolls, erfordert ein Echtheitszertifikat, das der Browser bei der ersten Verbindung vom Server

anfordert. In diesem Zertifikat steht auch der Zertifizierungspfad, mit dem die Abfolge der Zertifizierung definiert dargestellt ist. In der Regel akzeptiert der Browser Zertifikate von Zertifizierungsbehörden (siehe beispielsweise *http://www.trustcenter.de*) anstandslos. Solche Zertifikate kosten Geld und vor allem Mühe, da die Zertifizierungsbehörden die Echtheit des Administrators überprüfen und diese dann garantieren.

Erkennt der Browser die Zertifizierungsstelle oder das Zertifikat nicht, fragt er den Anwender, ob er das Zertifikat akzeptiert. Bei einer negativen Antwort wird die Verbindung in der Regel abgebrochen. In allen positiven Fällen wird die Verschlüsselungsart und -stärke ausgehandelt und die Verbindung damit aufgebaut.

Ein privat erstelltes Zertifikat reicht für private Anwendungen völlig aus. Es kann jederzeit durch ein öffentliches Zertifikat ersetzt werden, vielmehr muß das hier erzeugte Zertifkat sogar für die Bestellung des echten Zertifikats eingereicht werden.

Ein Wort zu den zu Zertifikaten

PEM (»Privacy Enhanced Mail«) ist ein offizieller Internet-Standard und in RFC 1421 bis 1424 beschrieben. Die Entwicklung von PEM wurde von einer Arbeitsgruppe der IETF (»Internet Engineering Task Force«) durchgeführt. Das Schlüsselmanagement ist strukturiert: die Schlüssel werden von Zertifizierungsbehörden (»Certification Authorities«) in Form von Zertifikaten bescheinigt. Jedes Zertifikat hat eine eindeutige Seriennummer. Die Zertifikate enthalten einen mit dem privaten Schlüssel der Zertifizierungsbehörde unterschriebenen MD5-Hash. Die Zertifizierungsbehörden müssen wiederum vom sogenannten PCAs (»Policy Certification Authorities«) zertifiziert werden, die schlußendlich von der IPRA (»Internet Policy Registration Authorities«) zertifiziert sind.

Die im folgenden erzeugten Zertifikate sind rein privater Natur und genügen nicht dem obigen Standard, da der Zertifizierungspfad nicht vollständig ist, dienen aber als Basis.

Es wird nun beschrieben, wie man unter NetBSD sehr einfach und schnell ein solches privates Zertifikat erzeugt und den (vorhandenen) Apache konfiguriert. Der Apache sollte bereits installiert und konfiguriert sein. Falls nicht, werden die folgenden Schritte dies aber gleich mit erledigen und dabei den Apache aus der NetBSD-Package-Distribution verwenden. Wichtig zur Vorbereitung der Installation ist die Kenntnis des Host- und Domainnamens des Servers, auf dem der Apache installiert wird. Ein vollständig eingerichtetes Domain System ist Voraussetzung – Sie müssen Ihren Server mit seinem Namen adressieren können (zum Beispiel *ping www.domain.de*).

Das NetBSD-Package »Apache-ssl« installieren

Das erforderliche Paket heißt *ap-ssl* und befindet sich im NetBSD-pkgsrc-Baum unter »www«. Das Binär-Paket heißt *ap-ssl-2.8.12* (oder aktueller). Dieses Paket installiert die freie Software *mod_ssl* und paßt sie direkt an NetBSD und den Apache an. Installieren Sie das Paket wie gewohnt mit *make install* oder *pkg_add*. Die Installation gibt am Ende eine Anleitung aus, die hier im folgenden übersetzt ist:

```
/usr/pkgsrc/www/ap-ssl

   [Folgende Dateien wurden installiert und dienen der Konfiguration von
mod_ssl:]

    /usr/pkg/etc/httpd/ssl.crl/Makefile.crl
    /usr/pkg/etc/httpd/ssl.crl/README.CRL
    /usr/pkg/etc/httpd/ssl.crt/Makefile.crt
    /usr/pkg/etc/httpd/ssl.crt/README.CRT
    /usr/pkg/etc/httpd/ssl.crt/ca-bundle.crt
    /usr/pkg/etc/httpd/ssl.key/README.KEY
    /usr/pkg/etc/httpd/ssl.prm/README.PRM

Sie müssen nun die Zertifikate und Schlüssel für Ihren Server erzeugen, bevor
der Apache mit mod_ssl gestartet werden kann. Bitte lesen Sie dazu die Datei

    /usr/pkg/share/doc/mod_ssl/README.mkcert

um mehr Informationen über die Vorgehensweise zu erfahren.


Die ssl-spezifischen Konfigurationsabschnitte der Standard "httpd.conf" des
Apache sind durch folgende Gruppierung gekennzeichnet:

  <IfDefine SSL>
  </IfDefine>

Um den Apache mit mod_ssl-Unterstüzzung zu starten, müssen Sie diesen mit
"apachectl startssl" statt "apachectl starten". Dies können Sie bei
Systemstart auch in der Datei "/etc/rc.conf" wie folgt hinterlegen:

        apache=YES
        apache_start="startssl"
```

Der Installationsvorgang hat also einige Dateien in »/usr/pkg/etc/httpd« angelegt, wo bereits der Apache seine Konfigurationsdatei »httpd.conf« sucht, die, wie oben angegeben, schon Informationen für das *mod_ssl* enthält. Das Zertifikat und die Schlüssel soll man nun nach Anleitung in der »README.mkcert« erzeugen.

Der Apache ist nun für *ssl* vorbereitet und kann wie beschrieben vom System automatisch gestartet werden oder durch händischen Aufruf der Startdatei »etc/rc.d/apache startssl«. Da aber die Zertifikate noch nicht erzeugt wurden, wird der Apache nicht verschlüsselt arbeiten.

Als nächstes ist nun das Programm *mkcert* aufzurufen, das mitinstalliert wurde. Im folgenden ein vollständiger Verlauf der Installation mit den einzelnen Benutzereingaben, die anschließend erklärt sind. Bitte lesen Sie dieses Kapitel einmal bis zu Ende durch, damit Sie die Tragweite der Eingaben verstehen. Die Ausgabe des Installationsproto-

kolls ist in englischer Sprache – lassen Sie sich davon nicht irritieren, die nötigen Eingaben werden erklärt.

```
root@desktop:~#/usr/pkg/sbin/mkcert -t custom -a RSA
SSL Certificate Generation Utility (mkcert.sh)
Copyright (c) 1998-2000 Ralf S. Engelschall, All Rights Reserved.

Generating custom certificate signed by own CA [CUSTOM]

-------------------------------------------------------------------

STEP 1: Generating RSA private key for CA (1024 bit) [ca.key]
3216138 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.........++++++
...++++++
e is 65537 (0x10001)

-------------------------------------------------------------------

STEP 2: Generating X.509 certificate signing request for CA [ca.csr]
Using configuration from /tmp/.mkcert.cfg
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
1. Country Name             (2 letter code) [XY]:DE
2. State or Province Name   (full name)     [Snake Desert]:NRW
3. Locality Name            (eg, city)      [Snake Town]:Koeln
4. Organization Name        (eg, company)   [Snake Oil, Ltd]:BSD-User GmbH
5. Organizational Unit Name (eg, section)   [Certificate Authority]:Team BSD
6. Common Name              (eg, CA name)   [Snake Oil CA]:www.domain.de
7. Email Address            (eg, name@FQDN)
[ca@snakeoil.dom]:webmaster@domain.de
8. Certificate Validity     (days)          [365]::

-------------------------------------------------------------------

STEP 3: Generating X.509 certificate for CA signed by itself [ca.crt]
Certificate Version (1 or 3) [3]::
Signature ok
subject=/C=DE/ST=NRW/L=Koeln/O=BSD-User GmbH/OU=Team
BSD/CN=www.domain.de/Email=webmaster@domain.de
Getting Private key
Verify: matching certificate & key modulus
```

```
read RSA key
Verify: matching certificate signature
/usr/pkg/etc/httpd/ssl.crt/ca.crt: /C=DE/ST=NRW/L=Koeln/O=BSD-User
GmbH/OU=Team BSD/CN=www.domain.de/Email=webmaster@domain.de
error 18 at 0 depth lookup:self signed certificate
OK
```

```
STEP 4: Generating RSA private key for SERVER (1024 bit) [server.key]
3216138 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
..++++++
..++++++
e is 65537 (0x10001)
```

```
STEP 5: Generating X.509 certificate signing request for SERVER [server.csr]
Using configuration from /tmp/.mkcert.cfg
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
1. Country Name              (2 letter code) [XY]:DE
2. State or Province Name    (full name)     [Snake Desert]:NRW
3. Locality Name             (eg, city)      [Snake Town]:Koeln
4. Organization Name         (eg, company)   [Snake Oil, Ltd]:BSD-User GmbH
5. Organizational Unit Name (eg, section)    [Webserver Team]:Team BSD
6. Common Name               (eg, FQDN)      [www.snakeoil.dom]:www.domain.de
7. Email Address             (eg, name@fqdn)
[www@snakeoil.dom]:webmaster@domain.de
8. Certificate Validity      (days)          [365]:
```

```
STEP 6: Generating X.509 certificate signed by own CA [server.crt]
Certificate Version (1 or 3) [3]:
```

```
STEP 6: Generating X.509 certificate signed by own CA [server.crt]
Certificate Version (1 or 3) [3]:
Signature ok
subject=/C=DE/ST=NRW/L=Koeln/O=BSD-User GmbH/OU=Team
BSD/CN=www.domain.de/Email=webmaster@domain.de
```

```
Getting CA Private Key
Verify: matching certificate & key modulus
read RSA key
Verify: matching certificate signature
/usr/pkg/etc/httpd/ssl.crt/server.crt: /C=DE/ST=NRW/L=Koeln/O=BSD-User
GmbH/OU=Team BSD/CN=www.domain.de/Email=webmaster@domain.de
error 18 at 0 depth lookup:self signed certificate
/C=DE/ST=NRW/L=Koeln/O=BSD-User GmbH/OU=Team
BSD/CN=www.domain.de/Email=webmaster@domain.de
error 7 at 0 depth lookup:certificate signature failure
```

```
STEP 7: Enrypting RSA private key of CA with a pass phrase for security
[ca.key]
The contents of the ca.key file (the generated private key) has to be
kept secret. So we strongly recommend you to encrypt the server.key file
with a Triple-DES cipher and a Pass Phrase.
Encrypt the private key now? [Y/n]:n
Warning, you're using an unencrypted private key.
Please notice this fact and do this on your own risk.
```

```
STEP 8: Enrypting RSA private key of SERVER with a pass phrase for security
[server.key]
The contents of the server.key file (the generated private key) has to be
kept secret. So we strongly recommend you to encrypt the server.key file
with a Triple-DES cipher and a Pass Phrase.
Encrypt the private key now? [Y/n]: n
Warning, you're using an unencrypted RSA private key.
Please notice this fact and do this on your own risk.
```

```
RESULT: CA and Server Certification Files

o  /usr/pkg/etc/httpd/ssl.key/ca.key
   The PEM-encoded RSA private key file of the CA which you can
   use to sign other servers or clients. KEEP THIS FILE PRIVATE!

o  /usr/pkg/etc/httpd/ssl.crt/ca.crt
   The PEM-encoded X.509 certificate file of the CA which you use to
   sign other servers or clients. When you sign clients with it (for
   SSL client authentication) you can configure this file with the
   'SSLCACertificateFile' directive.
```

```
o  /usr/pkg/etc/httpd/ssl.key/server.key
   The PEM-encoded RSA private key file of the server which you configure
   with the 'SSLCertificateKeyFile' directive (automatically done
   when you install via APACI). KEEP THIS FILE PRIVATE!

o  /usr/pkg/etc/httpd/ssl.crt/server.crt
   The PEM-encoded X.509 certificate file of the server which you configure
   with the 'SSLCertificateFile' directive (automatically done
   when you install via APACI).

o  /usr/pkg/etc/httpd/ssl.csr/server.csr
   The PEM-encoded X.509 certificate signing request of the server file which
   you can send to an official Certificate Authority (CA) in order
   to request a real server certificate (signed by this CA instead
   of our own CA) which later can replace the
/usr/pkg/etc/httpd/ssl.crt/server.crt
   file.

Congratulations that you establish your server with real certificates.
```

Sie sehen, daß die Eingaben zweimal getätigt wurden. Einmal für das eigene (Administrator-) Zertifikat und einmal für den Server. Das Zertifikat für den Server wird mit dem Zertifikat von Ihnen signiert.

Der Zertifikat-Erzeuger fragt Sie ein paar Daten ab. Für eine rein privat orientierte Nutzung ist es nur wichtig, daß der sogenannte »Common Name« genau dem Servernamen entspricht, für den Sie das Zertifikat erzeugen. Dabei muß ein sogenannter FQDN (Full Qualified Domain Name) verwendet sein, also Hostname und Domainname wie *www.domain.de* oder *desktop.domain.com*. Ein vollständig eingerichtetes Domain-System ist also Voraussetzung. Eine IP-Nummer kann nicht benutzt werden, dies würde auch dem Prinzip nicht entsprechen.

Folgende Daten werden abgefragt, geben Sie bitte keine Umlaute oder andere Sonderzeichen bei der Eingabe ein:

```
1. Country Name              (2 letter code) [XY]:DE
```

Dies ist das Kürzel des Landes, in dem der Beantrager registriert ist. Dies ist international – ähnlich wie bei KFZ-Zeichen – geregelt. Im deutschsprachigen Raum sind natürlich auch »AT« und »CH« verwendbar.

```
2. State or Province Name    (full name)      [Snake Desert]:NRW
```

Dies soll die Region oder der Kreis sein. Hier spätestens holen uns die amerikanischen Werte ein, die für Deutschland nicht übertragbar sind. Es ist aber erlaubt, diesen Wert leer zu lassen. Tun Sie dies mit einem ».« als Eingabe. »NRW«im Beispiel steht für »Nordrhein-Westfalen«.

```
3. Locality Name             (eg, city)       [Snake Town]:Koeln
```

Der Name der Stadt. Im Beispiel »Köln«.

```
4. Organization Name         (eg, company)    [Snake Oil, Ltd]:BSD-User GmbH
```

Der Name der beantragenden Firma oder Organisation. Hier im Beispiel eine fiktive Firma.

```
5. Organizational Unit Name (eg, section)    [Webserver Team]:Team BSD
```

Der Name der beantragenden Abteilung, was gerade bei großen Firmen und Organisationen wichtig ist. Als privater Benutzer können Sie auch einen leeren Wert mit ».« eintragen.

```
6. Common Name        (eg, FQDN)      [www.snakeoil.dom]:www.domain.de
```

Hier der besprochene FQDN-Name.

```
7. Email Address      (eg, name@fqdn) [www@snakeoil.dom]:webmaster@domain.de
```

Die offizielle Mail-Adresse des Ansprechpartner zu Fragen zum Zertifikat. Sie wird auch mit den persönlichen Benutzer-Zertifikaten abgeglichen, wenn diese vorhanden sind.

```
8. Certificate Validity      (days)           [365]:
```

Gültigkeitszeitraum. Nach einem Jahr muß das Zertifikat erneut angelegt werden. Ein Browser prüft dies.

Diese Daten werden einem Benutzer, der diesen Server demnächst anspricht, übermittelt. Er soll anhand dieser Werte entscheiden, ob er sensible Daten zu Ihnen übertragen möchte.

Wie beschrieben wurden nun die Zertifikate und Schlüssel in »/usr/pkg/etc/httpd/» abgelegt:

- */usr/pkg/etc/httpd/ssl.key/ca.key*

  Dies ist die PEM-verschlüsselte, private RSA-Schlüsseldatei, zugeteilt durch die Zertifizierungsbehörde. Sie wird zur Signierung anderer Server und Clients verwendet. Diese Datei hier wurde aber mit obigen *mkcrt* erzeugt und nicht durch eine Zertifizierungsstelle authentifiziert. Diese Datei – und schon gar nicht eine echte Version davon – darf man niemals herausgeben! Sie garantiert die Echtheit der folgenden Zertifikate. Wenn Sie diese Datei nicht schützen, kann Mißbrauch damit nicht ausgeschlossen werden.

- */usr/pkg/etc/httpd/ssl.crt/ca.crt*

  Dies ist das PEM-verschlüsselte X.509-Zertifikat, zugeteilt durch die Zertifizierungsbehörde. Es wird zur Signierung anderer Server und Clients verwendet. Wenn Sie Clients damit signieren, etwa zur SSL-Client-Authentifizierung, können Sie dieses Zertifikat in der Apache-Direktive *SSLCACertificateFile* angeben.

- */usr/pkg/etc/httpd/ssl.key/server.key*

  Diese PEM-verschlüsselte Datei ist ein privater RSA-Schlüssel des Servers, den Sie mit der Direktive *SSLCertificateKeyFile* im Apache angeben. Diese Datei – und schon gar nicht eine echte Version davon – darf man niemals herausgeben!

- */usr/pkg/etc/httpd/ssl.crt/server.crt*

  Diese Datei enthält das PEM-verschlüsselte X.509-Zertifikat für den Server, den Sie mit der Direktive *SSLCertificateKeyFile* im Apache angeben.

- */usr/pkg/etc/httpd/ssl.csr/server.csr*

  Diese PEM-verschlüsselte Datei enthält ein X.509-Zertifikat des Servers, das Sie zur Beantragung eines echten Server-Zertifikats zu einer Zertifizierungsbehörde senden können. Die dann erhaltene Datei wird unter »/usr/pkg/etc/httpd/ssl.crt/server.crt« abgelegt. Für diesen Vorgang ist es allerdings auch wichtig, daß die Ausgangsdatei »server.csr« mit einem echten privaten Schlüssel signiert ist. Das heißt, Sie müssen sich erst selbst persönlich zertifizieren lassen.

Als nächstes muß der Apache noch konfiguriert werden. Wenn Sie den Standard-Apache aus der NetBSD-Package-Sammlung verwenden, besitzt dieser bereits die notwendigen Konfigurationen und ein Neustart reicht aus, um neben einem *http://www.domain.de* auch ein *https://www.domain.de* zur Verfügung zu stellen. Weitere Einzelheiten entnehmen Sie bitte der Dokumentation des Apache, zu finden unter »/usr/pkg/share/doc/httpd«.

Wie bereits oben zitiert, starten Sie den Apache bei Systemstart per Konfigurationseintrag in der »/etc/rc.conf«:

```
apache=YES
apache_start="startssl"
```

Manuell geht der Start über das Skript */etc/rc.d/apache startssl*.

7.6 Dienste von TCP/IP

7.6.1 Installation des ftp-Servers

von Markus Illenseer

Der ftp-Server ist der Lastenesel unter den Internet-Diensten. Obwohl er etliche Jahrzehnte älter ist als ein Webserver, ist er dennoch weit verbreitet, da er einige Vorteile gegenüber einem Webserver bietet: Datei-Upload und Benutzermanagement.

Natürlich kann auch ein Webserver wie der Apache so konfiguriert werden, daß man einen Datei-Upload einrichten kann und Benutzermanagement kennt auch er. Aber für den Upload braucht man eine zusätzliche Applikation und das Benutzermanagement ist nicht direkt kompatibel mit dem Benutzermanagement des Betriebssystems.

Der mitgelieferte in der Basis-Installation enthaltene ftp-Server dagegen kann mit dem Benutzermanagement von NetBSD verwendet und kombiniert werden. Auch benötigt er in einer Grundversion keinen eigenen Benutzer und keine eigenen Dateistrukturen. Eigentlich lauter Vorteile, wenn da nicht der eklatante Nachteil des ftp-Protokolls selbst wäre, nämlich das Übertragen des Benutzernamens und Kennworts im Klartext in der Standardkonfiguration. Jedermann mit physikalischen Zugang im LAN des ftp-Servers – oder auf dem Weg dorthin – könnte diese Daten abfangen und die Sitzung sogar – mit viel Wissen – trickreich übernehmen.

Oft aber lassen einem komplexe Firewall-Konzepte keine andere Wahl, da es für *ssh*, *ssl* und andere in Frage kommende Protokolle keine Applikationsgateways gibt oder andere Probleme auftauchen. Wenn man sich an die Basis hält, kann eigentlich nicht viel passieren.

Start des ftpd

Der ftp-Server wird bei Bedarf durch den *inetd* gestartet. Der *inetd* wird beim Systemstart als einer der ersten Prozesse gestartet. In der Datei »/etc/inetd.conf« steht *ftp* auch gleich in der ersten Zeile:

```
#ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -ll
```

Das Doppelkreuz (»Number Sign« im Englischen) ist in der Standard-Installation von NetBSD eingetragen, die Zeile also kommentiert. Somit startet der ftp-Server nicht. Entfernen Sie das Doppelkreuz, startet der *inetd* bei Bedarf den ftp-Server *ftpd* mit der Option *-ll*. Was die einzelnen Einträge in der »inetd.conf« bedeuten, lesen Sie bitte beim *inetd*-Server nach. Wenn Sie Einträge ändern, müssen Sie den »inetd.conf« veranlassen, seine Konfiguration erneut einzulesen:

```
kill -HUP `cat /var/run/inetd.pid`
```

Der ftp-Server seinerseits kennt diverse Konfigurationsdateien die – als Basis-Systemdienst – alle in »/etc« beheimatet sind:

/etc/ftpchroot	Liste der (normalen) Benutzer, deren Benutzerverzeichnis mittels chroot(2) verändert werden soll.
/etc/ftpd.conf	Konfiguriert Datei-Umsetzungen und andere Einstellungen.
/etc/ftpusers	Enthält die Liste aller nicht erlaubten oder eingeschränkten Benutzer.
/etc/ftpwelcome	Willkommenstext vor dem Einloggen.
/etc/motd	Willkommenstext nach dem Einloggen.
/etc/nologin	Wenn diese Datei vorhanden ist, wird der Inhalt angezeigt, aber der Zugang ansonsten verweigert.

Weitere relevante Dateien sind:

/var/run/ftpd.pids-CLASS	Statusdatei für erfolgreiche eingeloggte Prozesse der ftp-Server-Klasse *CLASS*.
/var/run/utmp	Enthält die Liste aller derzeit eingeloggten Benutzer.
/var/log/wtmp	Login-Protokoll.

Nach einer frischen Installation von NetBSD und dem Entfernen des Doppelkreuzes in der »inetd.conf« und dem Neuladen der Konfiguration durch den *inetd* ist ftp nun sofort verfügbar. Für das Sofort-Aha probieren wir das gleich aus:

```
%> /etc/rc.d/inetd restart
%> ftp localhost
Trying 127.0.0.1...
Connected to localhost.
220-
220 localhost FTP server (NetBSD-ftpd 20020615) ready.
Name (localhost:joe):
331 Password required for joe.
Password:
230-
    NetBSD 1.6_STABLE (SERVER) #0: Mon Nov 25 19:34:55 CET 2002

    Welcome to NetBSD!

230 User joe logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Verlassen Sie den Client mit *bye*.

```
221-
    Data traffic for this session was 0 bytes in 0 files.
    Total traffic for this session was 554 bytes in 0 transfers.
221 Thank you for using the FTP service on localhost.
```

Das war die erste ftp-Sitzung mit Ihrem Server! Nun noch Erklärungen zu ein paar Einstellungen in den Konfigurationsdateien, um eine gewissen Basis-Sicherheit zu erreichen:

Die Datei »/etc/ftpusers« entscheidet, welcher System-Benutzer sich per ftp einloggen darf. Eine Beispieldatei befindet sich in »/usr/share/examples/ftpd/ftpusers«. Eines der allerwichtigsten Beispiele dort ist, daß die Systembenutzer *root* und alle Benutzer, die in der Gruppe *noftp* eingetragen sind, sich nicht einloggen dürfen. Der Eintrag in »/etc/ftpusers« sieht also zum Beispiel wie folgt aus:

```
root                    deny
*:noftp                 deny
```

Hinweis: Zwischen dem Wort *root* und dem Wort *deny* beziehungsweise zwischen *noftp* und *deny* werden ein oder mehrere {Tab}-Zeichen als Freizeichen verwendet. Ein einfaches Leerzeichen genügt nicht als Platzhalter!

Befindet sich also der Benutzer *joe* in der Gruppe *noftp* (siehe Datei »/etc/group«), kann er sich nicht mehr einloggen:

```
%> ftp localhost
Trying 127.0.0.1...
Connected to localhost.
220-
220 localhost FTP server (NetBSD-ftpd 20020615) ready.
Name (localhost:joe):
331 Password required for joe.
Password:
530 User joe may not use FTP.
ftp: Login failed.
ftp>
```

Alleine die Tatsache, daß der Benutzer *joe* in der Datei »/etc/group« in der Zeile für die Gruppe *noftp* eingetragen ist, reicht, daß er nicht mehr per ftp Zugang findet. Die Zeile in »/etc/group« sieht beispielsweise so aus:

```
noftp:*:205:joe
```

Weitere Benutzer, die sich nicht einloggen dürfen, werden einfach durch Komma getrennt hinter den Namen von *joe* angehängt.

Es geht auch anders herum. Wir verneinen explizit jedem den Zugang und öffnen ihn dann nur ganz bestimmten Benutzern. Die Datei »/etc/ftpusers« sieht dann wie folgt aus:

```
joe             allow
*               deny
```

Der Benutzer *joe* darf sich per ftp einloggen, aber niemand sonst, auch nicht *root*!

Bislang haben wir nur Benutzer erlaubt oder verweigert, die im System bekannt sind. Also Einträge aus »/etc/passwd«, die natürlich nur mittels *vipw* oder *usermod* verändert werden sollten. Was aber kennzeichnete den ftp-Server früher und auch noch heute? Der sogenannte »anonyme Benutzer«! Man kann sich auf einem ftp-Server so einloggen, daß man sich mit dem Benutzer *ftp* beziehungsweise *anonymous* ohne Passwort einloggt und dann in einem speziell eingerichteten Bereich Dateien lesen und – konfigurierbar – schreiben kann. Das ist interessant für den unkomplizierten Dateiaustausch und besonders für große Dateien geeignet, die sich nicht per E-Mail oder anders übertragen lassen. Denken Sie bitte immer daran, daß dieser Benutzeraccount kein Passwort kennt und jedermann weltweit die Dateien ansehen kann, die vom Vorgänger hochgeladen wurden, auch wenn das nicht eigentlich erwünscht ist. Wie man ihn einrichtet, erfahren Sie im anschließenden Kapitel.

Sie haben vielleicht schon bemerkt, daß ein Standardzugang zum ftp beim Einloggen bei einem Benutzer im Home-Verzeichnis endet. Sie können sich aber mit dessen Rechten bequem im gesamten System umsehen. Also auch im Verzeichnis »/etc« (mit all seinen Geheimnissen) oder im Verzeichnis »/home« (mit allen anderen Benutzern im System). Dies geschieht mit den Lese- und Schreibrechten des eingeloggten Benutzers – aber manchmal ist es nicht erwünscht, daß dies machbar ist. Beispielsweise bei bestimmten Benutzern wie *anonymous* oder weil Sie Ihren Benutzern per ftp einen direkten Schreibzugang in deren Webseite erlauben wollen, die in einem bestimmten Systempfad liegt, den der Benutzer nicht verlassen soll.

Dazu gibt es die Option *chroot* in der Datei »/etc/ftpusers«:

```
joe             allow               chroot
*               deny
```

Der Benutzer *joe* muß nun noch in der Datei »/etc/ftpchroot« definiert werden:

```
joe
```

(Pro Benutzer eine Zeile.)

Danach kann der Benutzer *joe* sich zwar ganz wie gewohnt einloggen. Aber alle Dateien oberhalb oder parallel zu seinem Home-Verzeichnis sind nicht mehr für ihn erreichbar. Es sieht für ihn so aus, als sei er in »/« eingeloggt. Sprich: sein Home-Verzeichnis wurde mittels *chroot* umdefiniert.

Wenn man nun nicht für jeden einzelnen User eine Zeile anlegen will, um dieses Verhalten für alle erlaubten Benutzer zu konfigurieren, macht man dies wie folgt: Zuerst wird die Datei »/etc/ftpusers« wieder in den normalen Zustand gebracht. Dort wird also jeder Benutzer eingetragen, dem ftp erlaubt wird. In der Datei »/etc/ftpchroot« werden alle Einträge entfernt. Nun wird die Datei »etc/ftpd.conf« angepaßt, indem die folgenden Einträge eingefügt werden. Bestehende Einträge sollten Sie bitte löschen, wenn sie vom Typ *chroot*, *homedir* und *umask* sind:

```
chroot chroot /home/%u
homedir chroot /
umask chroot 022
```

Damit kann nun jeder in »/etc/ftpusers« erlaubte Benutzer nicht mehr aus seinem Verzeichnis ausbrechen. Wenn also ein Hacker das Passwort ergaunert hat, kommt er nur bis dahin und nicht weiter. Dasselbe gilt auch für den Benutzer *anonymous* beziehungsweise *ftp*. Hier endet man im Home-Verzeichnis des in der Regel noch anzulegenden Benutzers *ftp* und dessen Home-Verzeichnis »/home/ftp«. Dieses Verfahren von *chroot* nennt man allgemein auch »Sandkasten« (weil man sich dort gefahrlos austoben kann).

Etwas komplexer geht es zu, wenn man Benutzer-Home-Verzeichnisse mit Benutzer-ftp-Verzeichnissen entkoppelt und den ftp-Benutzern eine eigene Spielwiese zuordnet.

Tragen Sie also zum Beispiel diese Zeilen in die »/etc/ftpd.conf« ein:

```
chroot chroot /home/ftp/%u
homedir chroot /data
umask chroot 022
```

Dann wird jeder in der Datei »/etc/ftpusers« mit der Option *chroot* eingetragene Benutzer in den Datenbereich »/home/ftp/<Benutzername>« umgelenkt. Dort ist der sichtbare Datenbereich das Verzeichnis »data«. Der Sandkasten ist komplett, wenn man folgende Hierarchie für jeden Benutzer einrichtet:

```
/home/ftp/joe
/home/ftp/joe/bin/
/home/ftp/joe/data
/home/ftp/joe/etc
/home/ftp/joe/usr
```

Weiterhin wird ein Softlink von »/home/ftp/joe/usr/bin« nach »/home/ftp/joe/bin« angelegt:

```
%> ln -s /home/ftp/bin /home/ftp/usr/bin
```

In den Verzeichnissen müssen folgende Dateien kopiert werden:

```
/home/ftp/joe/bin:
compress        aus         /usr/bin/compress
gzip            aus         /usr/bin/gzip
ls              aus         /bin/ls
sh              aus         /bin/sh
tar             aus         /usr/bin/tar
```

In »/home/ftp/joe/etc« wird eine Datei »group« mit dem folgenden Inhalt angelegt:

```
wheel:*:0:
ftpusers:*:203:
```

Außerdem findet hier eine neue eigenen Passwort-Datei »master.passwd« Platz:

```
root:*:0:0::0:0:Bastard Operator,,,:/:/dev/null
joe:*:127:203::0:0:joe user:/data:/dev/null
```

Nun werden eine Passwort-Datei und eine Shadow-Datei angelegt:

```
%> pwd_mkdb -p -d /home/ftp/joe master.passwd
```

Und noch eine eigene nur für *joe* bestimmte »motd«-Datei zur Begrüßung »geschrieben«:

```
echo "Hey Joe!" > motd
```

Das war's! Nun kann *joe* sich einloggen. Dabei wird sein in der Datei »/etc/passwd« hinterlegtes Passwort verwendet. Sein Datenverzeichnis »/data« (in Wirklichkeit »/home/ftp/joe/data«) ist für ihn schreib- und lesbar. Er kann mit ftp den Bereich nicht verlassen und auch nicht manipulieren. Selbst wenn er es schaffte, sich irgendwie die Rechte von *root* zu ergattern, er käme aus dem Sandkasten nicht heraus. Das Anlegen der Passwort-Datei ist nicht wichtig für das Passwort selbst – die Authentifizierung findet gegenüber der globalen Datei statt – sondern für die Gruppenrechte und Anzeige von *ls*, da der ftp-Daemon nun ja abgekapselt arbeitet und keinen Zugriff auf die globale Datei besitzt.

7.6.2 Konfigurieren eines Anonymous ftp-Servers

von Dr. Lex Wennmacher

Obwohl die Bedeutung des ftp-Dienstes durch das Aufkommen von HTTP zurückging, bieten viele Sites den ftp-Dienst an. Aus sicherheitsorientierter Sicht ist das ftp-Protokoll problematisch, weil zur Authentifizierung das Passwort im Klartext über das Netzwerk übertragen wird. Aus diesem Grund lehnen viele Administratoren den ftp-Zugang zu ihren Systemen über öffentliche Netzwerke ab, weswegen die Funktionalität

von ftp mehr und mehr durch SFTP und ssh/SCP ersetzt wird. Dieses Problem besitzt Anonymous ftp nicht, da keine Authentifizierung stattfindet. Richtig konfiguriert ist Anonymous ftp eigentlich ein ziemlich sicherer Dienst, solange der *ftpd*, der ja den Anonymous-ftp-Dienst zur Verfügung stellt, keine Sicherheitslöcher hat. Der in NetBSD integrierte *ftpd* wurde von Luke Mewburn entwickelt und ist auch für andere Unix-Varianten (beispielsweise Sun Solaris erhältlich). Bei seiner Entwicklung wurde besondere Sorgfalt auf große Sicherheit gelegt. Dieses Kapitel gibt eine Anleitung, wie man Anonymous ftp richtig konfiguriert, damit es dem Ansturm des Internet und den daraus resultierenden Anforderungen gewachsen ist. Die allgemeine Funktion des *ftpd* sowie dessen Einrichtung für lokale Benutzer wurde bereits im vorhergehenden Abschnitt beschrieben.

Zunächst legt man einen *ftp*-Account an, zum Beispiel mit *vipw* oder *useradd*. Sobald der *ftpd* eine Anonymous-ftp-Session angenommen hat, führt er einen *seteuid(2)*-Systemaufruf auf die UID des ftp-Accounts durch und arbeitet mit den Privilegien dieses Accounts weiter. Diese Privilegien sollten also so gering wie möglich gehalten werden. Im Passwort-Feld trägt man einen »*« ein, das verhindert, daß man sich via *login* als ftp einloggen kann. Bei der Wahl der UID und der GID ist man weitestgehend frei (UID 0 sollte aus naheliegenden Gründen allerdings nicht verwendet werden). Als Shell trägt man */sbin/ftplogin* ein, was ebenfalls bewirkt, daß der ftp-Account nicht als interaktiver Account benutzt werden kann. Da */sbin/ftplogin* nicht mit dem Betriebssystem ausgeliefert wird, erzeugt man es, indem man */sbin/nologin* kopiert oder linkt:

```
# cp /sbin/nologin /sbin/ftplogin
```

Außerdem muß man dafür Sorge tragen, daß */sbin/ftplogin* in »/etc/shells« verzeichnet ist. Dies ist erforderlich, damit ftpd den ftp-Zugang gewährt, beim Versuch eines interaktiven Zugangs erhält man jedoch keine brauchbare Shell.

Ein typischer Eintrag für ftp in »/etc/master.passwd« sieht also so aus:

```
ftp:*:60:60:Anonymous ftp Account:/home/ftp:/sbin/ftplogin
```

... und der Eintrag in »/etc/group«:

```
ftp:*:60:
```

Als Nächstes legt man die erforderliche Verzeichnis-Struktur an. Es ist sehr zu empfehlen, die via Anonymous ftp bereitzustellenden Dateien in einem eigenen Dateisystem zu halten (wir kommen später darauf zurück). Der benötigte Platz für die restlichen ftp-Dateien und -Verzeichnisse ist dann recht gering. Im folgenden Beispiel legen wir die ftp-Hierarchie unterhalb von »/home« an (lassen dieses Präfix jedoch dort weg, wo es möglich ist). Benötigt werden die Unterverzeichnisse »bin«, »etc« und »pub«:

```
# cd /home
# mkdir ftp
# mkdir ftp/bin
# mkdir ftp/etc
# mkdir ftp/pub
```

Bevor wir die Inhalte der Unterverzeichnisse besprechen, richten wir unser Augenmerk auf den Eigentümer und die Protections dieser Verzeichnisse. Ganz wichtig ist, daß keines dieser Verzeichnisse (noch der Dateien in diesen Verzeichnissen) dem Account *ftp* gehören, denn sonst könnte jeder Anonymous-ftp-Benutzer diese verändern. Ebenso darf die Gruppe des Eigentümers nicht *ftp* lauten. Es spricht nichts dagegen, den Eigentümer der Verzeichnisse auf *root* und die Gruppe auf *system* zu setzen. Wer besonders sicherheitsparanoid ist und deswegen die Zahl der von *root* besessenen Dateien minimieren will, der kann einen sonst nicht benutzten Extra-Account (und eine Extra-Gruppe) einrichten, beispielsweise *ftpadmin*, und diesem alle Verzeichnisse und Dateien übereignen. Prinzipiell ist aber nichts daran auszusetzen, für diesen Zweck *root* (und die Gruppe *system*) zu verwenden.

```
# chown root:system ftp
# chown root:system ftp/bin
# chown root:system ftp/etc
# chown root:system ftp/pub
```

Oder für die eben erwähnten Paranoiden:

```
# chown ftpadmin:ftpadmin ftp
# chown ftpadmin:ftpadmin ftp/bin
# chown ftpadmin:ftpadmin ftp/etc
# chown ftpadmin:ftpadmin ftp/pub
```

Die Protections der Verzeichnisse sollten so gesetzt sein, daß sie von niemandem beschreibbar sind:

```
# cd /home
# chmod 555 ftp/bin
# chmod 555 ftp/etc
# chmod 555 ftp/pub
# chmod 555 ftp
```

In »ftp/bin« kommen alle Programme zu liegen, die ftpd »on the fly« ausführen soll, beispielsweise *compress*, *gzip*, *tar*. Da *ftpd* als Bestandteil seines Sicherheitskonzeptes ein *chroot(2)* ausführt, um die Zugriffe des Anonymous-ftp-Benutzers auf Unterverzeichnisse des ftp-Verzeichnisses zu beschränken, stehen dem *ftpd* die Programme *compress*, *gzip* und *tar* aus »/usr/bin« nicht zur Verfügung und müssen daher als separate (statisch gelinkte) Kopien geführt werden. Für den Eigentümer der Programme in »ftp/bin« gilt

das oben gesagte: *root:system* (oder *ftpadmin:ftpadmin*). Bei den Protections sollte man ebenfalls sehr konservativ sein:

```
# cd /home
# chmod 111 ftp/bin/*
```

In »ftp« sollte ebenfalls ein Eintrag für »ftp/usr/bin« vorhanden sein (da *ftpd* Programme dort erwartet):

```
# cd /home
# mkdir ftp/usr
# chown root:system ftp/usr
# ln -s ftp/bin ftp/usr/bin
```

In »ftp/etc« kommen die Dateien zu liegen, die ftpd benötigt, um die Konversion von UIDs/GIDs auf reale Benutzernamen/Gruppennamen vorzunehmen. Es sind dies Kopien von »/etc/passwd«, »/etc/pwd.db«, »/etc/group«. Alle Dateien sollten nur lesbar sein:

```
# cd /home/ftp/etc
# cp /etc/passwd .
# cp /etc/pwd.db .
# cp /etc/group .
# chmod 444 passwd pwd.db group
```

Wichtig ist, darauf zu achten, daß in »ftp/etc/passwd« beziehungsweise in »ftp/etc/pwd.db« keine Paßwörter enthalten sind (wenn die Dateien wie oben angegeben angelegt werden, ist dies gewährleistet). Die Einträge werden nur gebraucht, um eine Übersetzung von UID/GID nach Namen vorzunehmen und nicht für irgendeine Form der Authentifizierung. Trüge man ein (verschlüsseltes) Passwort ein, wäre dies von jedem Anonymous-ftp-Benutzer zu lesen. Hacker könnten daraufhin versuchen, es mit *crack* oder ähnlichen Programmen zu dechiffrieren und sich damit Zugang zum System zu verschaffen.

Bei Bedarf kann man in »ftp/etc« auch noch eine Datei »motd« anlegen, deren Inhalt nach erfolgreichem Anonymous-ftp-Login dann vom *ftpd* angezeigt wird:

```
# cd /home/ftp/etc
# echo "Willkommen auf dem FOO-Server der BAR Gesellschaft" > motd
# chmod 444 motd
```

Wie bereits erwähnt, ist es sehr ratsam, ein eigenes Dateisystem für die unter »ftp/pub« abzulegenden Dateien zu verwenden. Einer der Gründe ist, daß ein Vollschreiben dieser Partition den Betrieb des ftp-Servers nicht gefährdet, ein anderer daß man das »ftp/pub«-Verzeichnis mit der Option *noexec* (beziehungsweise bei Bedarf *rdonly*) mounten kann. Ist es nicht erforderlich, daß die via Anonymous ftp angebotenen Dateien im

Betrieb geändert werden müssen, empfiehlt es sich, »ftp/pub« mit der Option *rdonly* zu mounten.

Das Verzeichnis »/home/ftp/tmp« wird nicht notwendigerweise benötigt. Falls es angelegt ist, benutzt ftpd es, um beim *LIST*-Befehl entstandene Fehlermeldungen abzuspeichern. Existiert es nicht, werden keine beim *LIST* entstandenen Fehlermeldungen an den Anonymous-ftp-Benutzer zurückgegeben, was nicht optimal ist. Die Gefahr, daß Anonymous-ftp-Benutzer Dateien hierher ablegen, kann man durch Benutzung des *upload*-Kommandos in »/etc/ftpd.conf« unterbinden. Falls man sich entscheidet, dieses Verzeichnis einzurichten, muß der Eigentümer (als Ausnahme von der obigen Regel) *ftp* lauten und die Protections sollten folgendermaßen gesetzt sein:

```
# cd /home
# mkdir ftp/tmp
# chown ftp ftp/tmp
# chmod 300 ftp/tmp
```

Eine Frage, die man sich als System-Administrator sehr gut überlegen sollte, ist, ob man es Anonymous-ftp-Benutzern gestattet, Dateien auf den ftp-Server hochzuladen. Wer sich nicht wirklich sicher ist, sollte dies nicht erlauben. Der Hintergrund ist, daß dieses Angebot regelmäßig von den sogenannten »Warez Kids« mißbraucht wird, um beispielsweise kopiergeschützte oder gehackte Software an andere weiterzuverbreiten. Man macht sich also schnell selbst einer Straftat mit schuldig. Die Warez-Gemeinde setzt eine Reihe von Scanning-Werkzeugen ein, um Anonymous ftp-Server zu finden, die ein Upload erlauben. Wir werden weiter unten aber ein Rezept angeben, wie man es ausgewählten Anonymous ftp-Benutzern dennoch erlauben kann, Dateien upzuloaden, ohne den Warez Kids Tor und Tür zu öffenen.

Mit den Konfigurationsmöglichkeiten, die »ftpd.conf« bietet und auf die wir weiter unten eingehen werden, kann man ebenfalls versuchen, die Warez Kids im Zaum zu halten, wenn man die Upload-Funktionalität benötigt. Das muß dann aber sehr sorgfältig erfolgen und getestet werden.

Entscheidet man sich dafür, allgemein die Upload-Funktionalität freizugeben, sollte man folgendes tun:

```
# cd /home/ftp
# mkdir incoming
# chown ftp:ftp incoming
# chmod 370 incoming
```

Wer es den Anonymous ftp-Benutzern erlauben will, die Inhalte von »ftp/incoming« zu sehen, der setze

```
# chmod 770 incoming
```

Im obigen Beispiel wird »incoming« als Unterverzeichnis von »/home/ftp« angelegt; auf einem realen ftp-Server ist es jedoch empfehlenswert, hierfür (analog zum »pub«-Verzeichnis) ein eigenständiges Dateisystem zu verwenden und dies (wiederum mit der Option *noexec*) nach »ftp/incoming« zu mounten.

Wer sich scheut, für anonyme ftp-Benutzer Uploads generell freizuschalten, kann auch folgenden Trick anwenden: In »ftp/pub« legt man ein Verzeichnis an, beispielsweise »incoming«, und setzt dessen Protections so, daß Benutzer *ftp* dessen Inhalt nicht lesen kann, wohl aber dieses Verzeichnis durchqueren (das heißt ausführen) kann:

```
# cd /home/ftp/pub
# mkdir incoming
# chown root:system incoming
# chmod 111 incoming
```

Nun kann man in diesem »incoming«-Verzeichnis ein weiteres (oder mehrere) Verzeichnisse anlegen, wobei man nicht offensichtliche Namen wählen sollte:

```
# cd /home/ftp/pub/incoming
# mkdir j9op3w
# chown ftp:ftp j9op3w
# chmod 755 j9op3w
```

Nun kann man den Namen dieses Verzeichnisses an Dritte weitergeben, die dann Dateien in dieses Verzeichnis ablegen können. Es dient also die Kenntnis des sonst nicht sichtbaren Namens des Verzeichnisses quasi als Passwort für den ftp-Upload-Dienst.

Somit wäre nun das ftp-Verzeichnis vollständig aufgesetzt und es bliebe noch die in »/etc/ftpd.conf« vorzunehmende Konfiguration des *ftpd* zu besprechen. Eine vollständige Beschreibung aller Möglichkeiten, die »ftpd.conf« bietet, würde den Rahmen dieses Buchs sprengen; die Manual-Seite von »ftpd.conf« leistet aber hier gute Dienste. Auf einige Einträge sei aber an dieser Stelle hingewiesen:

Entschließt man sich dazu, Uploads freizugeben, sollte man folgende Einträge in der Datei »/etc/ftpd.conf« vornehmen:

```
modify guest off
umask  guest 0707
```

Die *modify*-Anweisung unterbindet die Benutzung der Anweisungen *CHMOD*, *DELE*, *MKD*, *RMD*, *RNFR* und *UMASK* durch den Anonymous-ftp-Benutzer (er kann also zum Beispiel keine Unterverzeichnisse anlegen und seine *umask* nicht verändern). Durch das Setzen der *umask* auf 0707 wird erreicht, daß der Anonymous ftp-Benutzer selbst nicht mehr auf die von ihm hochgeladenen Dateien zugreifen kann. Falls es sinnvoll erscheint, kann man mit Hilfe der Anweisung *maxfilesize* die maximale Länge der hochzuladenden Dateien begrenzen:

```
maxfilesize all 2m
```

begrenzt die maximale Größe von Upload-Dateien auf zwei Mbyte.

Entschließt man sich dazu, Anonymous ftp-Uploads generell zu unterbinden, erreicht man dies mit dem Eintrag von

```
upload guest off
```

in der »/etc/ftpd.conf«. Dies unterbindet ebenfalls den oben erklärten Trick über den geheimen Dateinamen, da es generell die Benutzung der Befehle *APPE, STOR, STOU FTP* unterbindet.

Wie bei den meisten Unix-Netzwerk-Diensten muß man sie zur Inbetriebnahme in »/etc/inetd.conf« aktivieren, indem man das Kommentarzeichen »#« für den bereits vorhandenen Eintrag für *ftp* entfernt und dann ein *HUP*-Signal an *inetd* schickt.

7.6.3 Mit NTP die Zeit im Griff

von Thorsten Lindloff

PC-Uhren sind bekanntermaßen ungenau. Insbesondere im Netzwerk kann man sich Zeitabweichungen aber nicht leisten. Es bietet sich deshalb an, die Uhrzeit automatisch zu synchronisieren, entweder über einen Zeitserver im Internet oder über das DCF-77-Signal und eine Funkuhr.

Anno 1845 zog ein Schotte namens Sanford Fleming nach Kanada, um sich dort zum Ingenieur ausbilden zu lassen. Ab 1863 war Fleming an einem der größten Eisenbahnprojekte der Welt beteiligt: dem Bau des Canadian Pacific Railways, die die erste transkontinentale Eisenbahnverbindung Kanadas werden sollte. Diese Aufgabe warf für Fleming, der Chefingenieur in diesem Projekt war, auch ein anderes Problem auf, genaugenommen drehte es sich um die Zeitfrage. Der Kunde sollte sich schließlich auf das verlassen können, was auf den Fahrplänen steht.

In Rußland löste man das Problem einfach und pragmatisch. Man stellte alle Uhren auf den Bahnhöfen der ab 1891 entstandenen Transsibirischen Eisenbahn auf die Moskauer Zeit.

Das Gegenteil davon geschah in Deutschland. Wer von Nürnberg nach Berlin fuhr, mußte seine Uhr regelmäßig um sieben Minuten nachstellen. Das war übrigens auch im Kaiserreich noch so und sorgte vermutlich des öfteren für Flüche bei den Eisenbahnern und deren Kunden. Ein derartiges Gefummel war nichts für einen in Menschen wie Fleming. Als Bürger des britischen Empires dachte er eben in globalen Dimensionen. Fleming war einer der Initiatoren der Washingtoner Konferenz, in der 1884 der Nullmeridian in Greeenwich und die heutigen Zeitzonen definiert wurden.

Diese definierte Zeit wurde auf der Grundlage der Erdrotation festgelegt, nach der eine Erdumdrehung 86400 Sekunden dauert und nach der ein Tag sich in 24 Stunden aufteilt. Diese Definition sollte bis in unsere Tage hinein ihre Gültigkeit behalten. Bei

diesem 86400-sten Teil eines Tags handelt es sich um die sogenannte »Weltzeitsekunde«. Diese Sekunde war bis 1956 die offizielle Zeiteinheit.

Was man in den fünfziger Jahren des zwanzigsten Jahrhunderts nach Erfindung der Quarzuhr herausfand, ist der Umstand, daß die Erde sich nicht etwa gleichmäßig dreht, sondern unrund ist wie ein alter Autoreifen. Dazu kommt noch, daß die Rotationsgeschwindigkeit der Erde im langzeitigen Mittel abnimmt, womit auch die Vierundzwanzigstunden-Zeitspanne eines Tages sich verlängert.

Bereits 1956 hat man die Epheremidensekunde, die dem Mittelwert einer Weltzeitsekunde aus den Jahren 1700 bis 1900 entspricht; als Standard festgelegt, weil man glaubte, daß diese Sekunde genauer ist. Damals entschloß man sich, die Zeiteinheit über eine atomare Naturkonstante zu ermitteln. 1967 definierte man ein Zeitintervall als Atomsekunde, das an diese Epheremidensekunde angepaßt wurde und damit geringfügig kürzer ist als die heutige Weltzeitsekunde.

Am 1. Januar 1972, dem Datum, das den Anfangspunkt der UTC-Skala definiert, wurde die Differenz zwischen beiden Zeiten mit zehn Sekunden definiert.

Hier liegt bereits der wesentliche Unterschied zwischen der UTC und der Unix-Zeitrechnung: Die Unix- Zeitrechnung beginnt am 1. Januar 1970. Damit kennen Sie auch schon den Hauptgrund für den Umstand, daß ein Unix-Kernel keine Schaltsekunden kennt. Der andere Faktor ist das besagte »Eiern« der Erde: Die Tage sind nicht genau gleich lang.

In Deutschland ist alles geregelt, auch die Zeit. Sie wird hierzulande von der Physikalisch-Technischen Bundesanstalt in Braunschweig verwaltet, die zu diesem Zweck vier Caesium-Atomuhren betreibt, deren rechnerisches Mittel die gültige Zeit darstellt, mit der Sie Ihren Rechner auf mehreren Wegen synchronisieren können.

DCF-77

Das Signal, das der in Mainflingen bei Frankfurt stehende Sender verbreitet, sieht so aus: DCF-77 ist ein Langwellensender, der ein amplitudenmoduliertes Signal auf einem 77,5 kHz-Träger sendet. Die Amplitude wird bei Beginn einer neuen Sekunde auf 25 Prozent ihrer Stärke abgesenkt und bleibt auf diesem Wert hundert oder zweihundert Millisekunden stehen, um dann für die nächsten 800 oder 900 Millisekunden wieder auf ihre volle Stärke zu gehen. In der neunundfünfzigsten Sekunde jeder Minute wird auf die Absenkung des Signals verzichtet. Wird ein fallendes Signal erst nach 1800 Millisekunden empfangen, bedeutet das den Beginn eine neuen Minute. Durch den Abfall der Flanken bei jeder neuen Sekunde kann man sich mit Hilfe dieses Signals mit dem Sekundentakt synchronisieren. Zusätzlich wird in jeder Sekunde ein Bit mit einer Information übertragen. Wenn die Absenkung hundert Millisekunden dauert, handelt es sich um eine 0; ansonsten um eine 1. Daraus ergibt sich, daß pro Minute 59 Bit an Informationen übertragen werden. Darin sind Informationen wie Jahr, Monat, Tag, Stunde, Minute, Sekunde und Informationen wie die Sommerzeit und Schaltsekunden enthalten.

In einem Computer gibt es immer zwei Uhren. Die eine ist die eingebaute Hardware-Uhr und die andere die Software-Uhr, die mehr oder weniger im Betriebssystem-Kernel integriert ist und wesentlich genauer als die Hardware-Uhr geht, abgesehen von den Schaltsekunden.

Die Hardware-Uhr

In jedem modernen PC ist eine Uhr eingebaut, die batteriegepuffert und meistens mit einem eigenen Chip im CMOS-RAM integriert ist. Diese Uhr gibt die Zeit als Jahr, Monat, Tag, Stunde, Minute und Sekunde aus und wird meistens als *Real Time Clock* (RTC) bezeichnet. Als Identifikationshilfe: Hergestellt werden diese Chips meistens von Motorola oder Dallas Semiconductor.

Diese Uhren sind allerdings mit einigen Nachteilen behaftet:

- Sie lösen nur auf die volle Sekunde genau auf.
- Meistens werden aus Konstengründen Billigstquarze verwendet.
- Es gibt keine Möglichkeit, die Ganggeschwindigkeit mit einer externen Referenzuhr nachzuregeln.
- Eine solche Uhr weiß nicht, wo sie sich befindet, was zu einem Problem werden kann, wenn die Uhr auf die Ortszeit eingestellt ist.
- Einige dieser Uhren kennen zwar den Unterschied zwischen Sommer- und Winterzeit, diese aber nur nach den Vorgaben aus den USA, die von denen in Europa abweichen.

Viele Computer-Anwender stellen die Hardware-Uhr auf die Ortszeit ein. Auch das kann eine Ursache von Problemen sein. Spätestens dann, wenn auf einem Rechner mehr als ein Betriebssystem seinen Dienst tut, funktioniert das automatische Umstellen der Uhr nicht mehr wirklich, besonders, wenn das Programm die Hardwareuhr nachstellt. Bei dieser Vorgehensweise gibt es nämlich ein kleines Problem. Sie definieren bekanntlich in einer Konfigurationsdatei Ihre Zeitzone. Dann erfolgt die Umstellung auf die Sommerzeit. Das gerade gebootete System stellt die Zeit korrekt um. Sie starten nach dieser Umstellung den Rechner mit einem anderen Betriebssystem neu. Auch hier ist die Zeitzone nebst dem Wechsel auf Sommer- und Winterzeit definiert. Was macht das System? Genau – es stellt die Uhr nach. Wenn Sie jetzt das Ganze noch mit einem dritten Betriebssystem machen, ist die Konfusion perfekt: Sie sind der Zeit bereits um zwei Stunden vorausgeeilt.

Um so einem Problem aus dem Weg zu gehen, gibt es nur eine Lösung: Die Einstellung der Uhr auf eine Zeitzone, in der derartige Wechsel nicht stattfinden. Eine solche Zeit ist die UTC, die auch in der Schiffahrt und im Luftverkehr verwendet wird. Allerdings haben hier wiederum einige Betriebssysteme einer großen Firma aus Redmond ein mittelkleines Problem.

Zu Zeiten der Erfindung von Unix hat das noch niemanden wirklich interessiert. Es gab ja nur das kleine Arpanet, an das längst noch nicht jeder angeschlossen war.

Die Synchronisation

Man kann es drehen und wenden, wie man will: Wenn man keine einigermaßen genaue Zeitsynchronisation hätte, würde vieles nicht richtig funktionieren. Trotzdem wird der Synchronisation der Rechner nicht immer die Aufmerksamkeit geschenkt, die sie eigentlich verdient.

Wegen dieser kleinen Ungenauigkeiten der RTC besitzen Unix-Betriebssysteme, wie einige andere auch, eine eigene Software-Uhr. Um die Zeit zu ermitteln, wird regelmäßig ein Interrupt ausgelöst, der das Betriebssystem veranlaßt, die Zeit weiterzuzählen oder laufende Prozesse zu verwalten.

Auch die Unix-Zeit ist standardisiert; wie bereits vorhin erwähnt, beginnt die Unix-Zeitrechnung zwei Jahre vor der UTC. Die Zeitrechnung in Unix ist im Posix-Standard dokumentiert, in dem auch festgelegt ist, daß die Schaltsekunden, die bei den Atomuhren regelmäßig eingelegt werden, nicht mitgezählt werden.

Dieses Verfahren hat den Vorteil, daß jeder Tag eben besagte 86400 Sekunden lang ist. Aber auf der anderen Seite gibt es eben Zeiten, die nicht dargestellt werden können. Gemeint sind hier die Schaltsekunden: 1972-06-20 23:59:60 UTC ist so eine Uhrzeit.

Die Sekunden werden als vorzeichenbehafteter 32 Bit großer signed long int gezählt, um auch Zeiten ermitteln zu können, die vor dem 02. Januar 1970 lagen.

Zeit	Wert
1970-01-01 00:00:00 UTC	*0.000000*
1970-01-01 00:00:01 UTC	1.000000
1970-01-02 00:00:00 UTC	86400.000000
1969-12-31 23:59:59 UTC	-1.000000
1999-06-18 12:13:14 UTC	929707994.000000

Tabelle 7.5: Darstellung der Zeiten in Posix

Das letzte Beispiel in Tabelle 7.5 ist exemplarisch für die Nichtbeachtung der Schaltsekunden in Posix: Tatsächlich gab es seit 1972 bereits 22 Schaltsekunden. Korrekt wären 929708016 Sekunden, die seither vergangen sind.

Zur Hardware-Uhr

In ziemlich jedem Computer gibt es eine Hardware-Uhr. Auf den Intel-Plattformen »tickt« sie mit einer Frequenz von etwa hundert Hertz. Gesteuert wird diese Routine von einem Intel-Chip der Typen 8253 oder 8254. In diesem Chip ist ein 16-Bit-Zähler integriert, der mit einer Frequenz von 1,139180 MHz abwärts zählt. Wenn dieser Counter bei 0 angekommen ist, wird der Chip wieder auf einen Anfangswert gesetzt, ein Interrupt ausgelöst und ein neuer Countdown gestartet.

Dieser Anfangswert ist so definiert, daß dabei immer 100 Hz Interruptfrequenz herauskommen: 1139180/100 = 11391,80 oder 11392 Hz.

Daraus ergibt sich, daß die Uhr nicht genau mit 100 Hz tickt, sondern nur mit 99,998244 Hz. In Wirklichkeit wird also die Routine alle 10,000176 ms aufgerufen.

Diese Interrupt-Routine sieht so aus:

```
tick=10000
xtime.tv_usec += tick
if (xtime.tv_usec >= 1000000) {
  xtime.tv_usec -= 1000000;
  xtime.tv_sec++;
}
```

Eine weitere Variable wird im Kernel gehalten, um die Ungenauigkeiten auszugleichen. In dieser Variable werden die Ungenauigkeiten aufaddiert, bis die Abweichung eine Mikrosekunde beträgt. Um diesen Faktor wird dann der Zeitzähler korrigiert. Die Ungenauigkeit der Uhr beträgt so nie mehr als eine Mikrosekunde. Gleiches gilt für die Frequenz der Uhr. Auch hier ist die Abweichung niemals höher als eine Mikrosekunde pro Tick.

Das Ergebnis daraus ist eine Software-Uhr, die immer zu den Zeiten des Interrupts einen bereits sehr genauen Wert enthält. Der Nachteil ist die Schrittweite der Darstellung. Sie beträgt bei PCs immer 10000 Mikrosekunden und wird zwischen den Interrupts nicht verändert.

Das Stellen der Uhr

Wenn man ein neues Betriebssystem auf einem Rechner installiert, muß die Entscheidung getroffen werden, ob die Hardware-Uhr auf die lokale Zeit oder auf die UTC eingestellt wird. Wie schon beschrieben, ersparen Sie sich bei der UTC den Wechsel zwischen Sommer- und Winterzeit. Das bedeutet, daß die Uhr nie sich ändernden Gegebenheiten angepaßt werden muß.

Die Zeitzone richtet man unter NetBSD manuell oder mit dem Systeminstallationsprogramm zum Schluß der Installation ein.

Es wird ein symbolischer Link auf eine Datei in »/usr/share/zoneinfo« gesetzt. Wenn Sie das System mit einem Tool verwalten, sollten Sie auch die Zeitzone damit setzen. Es kann sonst passieren, daß sie bisweilen Ihre Informationen verlieren, weil das Tool diese ohne Vorwarnung überschreibt, was schon des öfteren für Verdruß gesorgt hat:

```
# ln -sf CET /usr/share/zoneinfo/[Ihre Lieblingsstadt oder CET]
```

Anschließend stellen Sie die Kerneluhr:

```
# date -s "May 07, 2002 09:30:00"
```

Geben Sie hier die lokale Zeit an. Sie können aber die Uhr mit dem Parameter *-u* auch nach der UTC stellen:

```
# date -s -u "May 07, 2002 09:30:00"
```

Wenn Sie das getan haben, stellen Sie die Hardware-Uhr mit *hwclock*. Das Programm liest die Zeit aus der Kerneluhr und überträgt sie, wie der Kommandoname schon sagt, auf die CMOS-Uhr im Rechner:

```
# hwclock -w
```

Oder mit UTC:

```
# hwclock -wu
```

Das Network Time Protocol

Entwickelt wurde das NTP-Protokoll von David Mills an der University of Delaware. Beschrieben ist es in RFC 1305. Interessant wird dieses Verfahren, sobald mehr als ein Rechner im Netzwerk installiert ist.

Setzt man ein Netzwerk-Dateisystem wie NFS ein, werden von dem Rechner, auf dem die Daten liegen, sogenannte »Time Stamps« erzeugt. Wurden diese Time Stamps von einem Rechner gelesen, dessen Uhr vorgeht, interpretiert dieser Rechner sie als falsch, weil sie aus seiner Sicht in der Zukunft liegen, was ja nicht sein kann.

Bei den Logdateien sieht das so aus: Jeder Rechner erzeugt solche Dateien. Wenn jede Uhr im Netzwerk anders geht, können Sie die Daten nicht miteinander vergleichen, weil sich die Ereignisse nicht genau zuordnen lassen.

Allein schon diese beiden Umstände erfordern zumindest eine Synchronisation der Uhren untereinander oder, was eben besser ist, mit einer genauen Referenz.

Das geschieht mit dem System-Aufruf *adjtime* oder dem moderneren *adjtimex*. Damit übermittelt man dem Kernel eine Variable, die ihm mitteilt, wie ungenau seine Uhr jetzt wirklich geht. Wenn sich herausstellt, daß die mit 10000 Ticks pro Sekunde laufende Uhr ungenau geht, wird die Frequenz je nach Bedarf um fünf Mikrosekunden erhöht oder gesenkt. Anhand der in *adjtime* übergebenen Werte kann der Kernel sich dann ausrechnen, wie lange es dauert, bis die Differenz wieder ausgeglichen ist.

Dargestellt werden alle Zeiten in diesem Protokoll als 64 Bit lange Zeichen mit 32 Bit vor und 32 nach dem Komma. Die errechnete Zeit ist die Zeit in Sekunden, die seit dem 01.01.1900 vergangen ist. Das bedeutet, daß bei Einführung der UTC bereits 2272060800,0 Sekunden vergangen waren, wenn man annimmt, daß es bis dahin keine Schaltsekunden gab.

Aufgrund der 32 Nachkommastellen ist die Auflösung der Zeitdarstellung von NTP 2^{32}.

Der Aufbau des Zeitsystems

Die Zeitsynchronisation funktioniert, wie fast alle Meßverfahren, nach einer hierarchischen Struktur. Bei diesem Protokoll gibt es eine solche Hierarchie aus mehreren Servern, die aus maximal elf Stufen bestehen kann.

Das erste System in dieser Hierarchie wird immer *Stratum 0* genannt. Stratum 0 bezeichnet die eigentliche Quelle der Zeit, genaugenommen ist das in Deutschland die Uhr in Braunschweig. Das bedeutet, daß die Rechner der PTB, die direkt mit diesen Uhren verbunden sind, bereits Stratum-1-Server sind. Alle nachfolgenden Systeme in dieser Hierarchie werden mit einer höheren Nummer versehen, die der Benutzer des lokalen Clients unter Beachtung der Hierarchie selbst vergeben kann. Die niedrigste Stufe und damit gleichzeitig ungenaueste ist Stratum 10. Von einem Stratum-10-Server kann die Zeit nicht mehr an eine niedrigere Stufe weitergereicht werden.

Zwei Server, die man als Stratum-1-Server betrachten kann, sind die Maschinen, die bei der PTB in Braunschweig ihre Zeit der Öffentlichkeit zur Verfügung stellen. Solange man die Zeit nicht zu kommerziellen Zwecken abfragt, ist der Dienst kostenlos. Ansonsten benötigt man eine Genehmigung. Die Server heißen »ntp1.ptb.de« und »ntp2.ptb.de«.

Wenn man die Rechneruhr nur synchronisieren will, reicht diese Pipe:

```
# ntpdate ntp1.ptb.de | hwclock -s
```

Damit erledigen Sie mehrere Einstellungen in einem Stück. Es handelt sich einmal um die Synchronisation der Softwareuhr und zum anderen um die Einstellung der Hardwareuhr. Wenn Sie sich per *ppp* ins Internet einwählen und nur über einen Rechner verfügen, sollten Sie das Kommando an das Ende des Einwahlskripts hängen, damit der Server unmittelbar nach dem Verbindungsaufbau abgefragt wird. Bei einer Flatrate bietet es sich vielleicht auch an, das mittels eines Cron-Jobs zu erledigen, der zuerst eine Verbindung ins Internet aufbaut, sich dann synchronisiert und am Schluß der ganzen Aktion die Zeit an die anderen Maschinen durchreicht. Es gibt der Möglichkeiten viele: Ich benutze einen alten Rechner als Router, der zuerst hochgefahren wird und einen Arbeitsrechner, der erst läuft, wenn die Verbindung ins Internet steht. Der Arbeitsrechner gibt bei mir den Server nach der Konfiguration weiter unten. Der Arbeitsrechner dient bei mir dann als Stratum-2-Server und sendet per Broadcast die Zeit an die anderen Maschinen im privaten Netz, einschließlich des Routers.

Besser bedient sind allerdings Anwender, die einen Server an einer richtigen Standleitung haben, die nicht dauernd zusammenbricht. Damit besteht die Möglichkeit, einen richtigen eigenen Server aufzusetzen, dessen Hierarchiestufen selbst definiert werden. Weil Ihr Server aber eine Maschine ist, die sich mit einer anderen synchronisiert, sollten Sie mit Stratum 2 anfangen. Am einfachsten ist es, wenn Sie einen Rechner in Ihrem Netz zum Server erklären, der die Zeit bei einem anderen Server abfragt und sie per Broadcast an die Clients im Netz verteilt. So eine Konfiguration sieht wie folgt aus. In die »/etc/ntp.conf tragen Sie ein:

```
server ntp1.ptb.de
server ntp2.ptb.de
fudge 127.0.0.1 stratum 2

broadcast 192.168.1.0
```

Als Server können Sie bis zu drei Maschinen angeben. Damit konfigurieren Sie die beiden NTP-Server bei der PTB als die Maschinen, mit denen Ihr lokaler Server sich synchronisieren soll. Das Kommando *fudge* bei 127.0.0.1 besagt nichts anderes, als daß diese Maschine eine Quelle ist, auf die kein hundertprozentiger Verlaß ist. *broadcast* verteilt die Zeit im Netz.

Bei den Clients muß folgendes in der »ntp.conf« stehen:

```
peer 192.168.1.1
broadcastclient
```

Nur der Vollständigkeit halber: Es geht natürlich auch mit den Rechnernamen.

Dann muß die maximale Abweichung vom lokalen Server konfiguriert werden. Das wird in der Drift-Datei getan. Der Inhalt der Datei ist eine einfache reale Zahl, die das Maximum der Abweichung in Sekunden ausdrückt. Auf NetBSD-Systemen finden Sie die Drift-Datei unter »/var/db«. Dort heißt sie »ntp.drift«:

```
0.05
```

Wenn Sie den *ntpd* jetzt auf allen Maschinen starten, sollten sich die Softwareuhren automatisch synchronisieren, was eine ganze Weile dauern kann. Nach einem Reboot der Systeme oder einem manuellem ntpd-Start funktioniert das Ganze recht zuverlässig.

Dann brauchen Sie nur noch mit *date* und *hwclock* die Hardwareuhr anzupassen und sollten ein leidiges Problem weniger haben.

Empfang der Zeit per Funk

Wenn das System nicht dauernd online ist und Sie auch keine Flatrate benutzen oder Sicherheitsbedenken haben, sollten Sie sich vielleicht eine eigene Uhr anschaffen. Zu diesem Zweck brauchen Sie eine entsprechende Funkuhr und meistens einen Treiber. Eine der preiswertesten Varianten ist eine passive Uhr der Firma Sure, die an den seriellen Port angeschlossen wird und mit einem Linux-Treiber kommt. Die passiven Uhren haben allerdings auch einen Nachteil: Wenn der DCF77-Sender ausfällt, fällt auch die Uhr aus. In solch einem Fall wird die Hardwareuhr des Rechners zur Zeitermittlung herangezogen. Für Einsatzzwecke in einer Verwaltung sollten diese Uhren allerdings hinreichend sein, wenn Sie sicher sind, daß die PC-Uhr nicht zu ungenau ist.

Wenn Sie in einem kritischen Bereich wie Anwendungen in der Meß- und Regeltechnik arbeiten, sollten Sie sich den vermeintlichen Luxus einer aktiven Uhr leisten, die auch bei Ausfall von DCF77 noch recht genau geht. Die Konfiguration beider Uhren ist in der Praxis nahezu identisch.

Bei aktiven Uhren ist darauf zu achten, daß die Quarze temperaturempfindlich sind. Diese Temperatur wird bei diesen Uhren gemessen und das Abdriften der Quarze regeltechnisch so weit reduziert, daß die Uhr immer hinreichend genau geht. Funkuhren lesen die Zeit per Mittelwelle oder GPS. Bei diesen Uhren kommen allerdings einige Sicherheitsmechanismen ins Spiel, die zumindest über eine gewisse Zeitspanne sicherstellen, daß die Zeit nicht allzu weit vom Standard abweicht. Aktive Uhren besitzen einmal die DCF77-Uhr, dann eine Quarzuhr und zusätzlich noch eine Notuhr. Wenn Sie ein System in Betrieb nehmen, wird die Zeit aus der Notuhr in die Quarzuhr übernommen, sofern die Uhr nicht länger als drei Tage ohne Strom war. Sobald der Empfang des DCF77-Signals möglich ist, werden die Quarz- und die Notuhr mit der aktuellen Zeit versorgt. Nach einiger Zeit weiß die Uhr, wie ungenau die Quarzuhr geht und korrigiert sie entsprechend. Wenn dann die Uhr in Mainflingen ausfällt, wird auf Quarzuhrbetrieb umgeschaltet. Diese Uhren können bis zu drei Tage ohne nennenswerte Abweichungen von der DCF77-Zeit »leben«.

Damit haben Sie noch einen weiteren wichtigen Grund für auftretende Ungenauigkeiten bei den eingebauten Hardwareuhren in den Computern.

Funkuhren und Windows-Systeme

Die traurige Wahrheit ist leider auch, daß viele Funkuhren nur für Windows-Systeme angeboten werden. Allerdings gibt es hier Abhilfe. Es gibt eine ntpd-Version, die auch ohne den Cyrus-Layer zumindest auf Windows NT und 2000 zum Laufen zu bekommen ist. Das bedeutet aber Lizenzkosten. Interessant sind deshalb Uhren, die wenigstens mit Linux laufen. Wenn Sie einen ntpd vom Linux-Distributor installiert haben, sollten Sie ihn aus zwei Gründen vom System räumen: Der eine ist der Umstand, daß Sie nicht wissen, welche Treiber in das System einkompiliert sind. Der andere Grund ist, daß Sie sicher sein können, daß die Version auf Ihrem System nach meinen Beobachtungen meistens schon recht alt ist. Das können Sie besser haben. Den ntpd bekommen Sie bei ziemlich allen bekannten Quellen und von einigen kommerziellen Anbietern. Die bekannteste ist wohl »www.ntp.org«.

Wenn Sie den ntpd heruntergeladen haben, entpacken Sie ihn mit diesem Kommando:

```
# ./configure --with-[Uhrenhersteller] && make && make install
```

Eine Liste der unterstützten Uhren finden Sie auf *http://www.eecis.udel.edu/~ntp/ntp_spool/html/refclock.htm*.

Die Konfiguration der Uhr in »/etc/ntp.conf« folgt immer demselben Muster: Zum einen wird festgelegt, wo synchronisiert werden soll und zu anderen in welcher Form. Hier ist ein Beispiel für eine Billiguhr, die oft auch bei ebay zu bekommen ist:

```
logfile /var/log/ntp
```

... ist selbstredend

```
server 127.127.35.0
```

Hier ist vor allem die dritte Zahl in der IP-Adresse wichtig. Sie teilt dem ntp mit, von welchem Uhrentyp er seine Daten abfragen soll. In diesem Fall handelt es sich um eine passive Uhr, die an den seriellen Port der Maschine angeschlossen ist.

```
server 127.127.1.0
```

Das ist die Hardware-Uhr des PCs, auf dem der Server läuft.

```
broadcast 192.168.1.255
```

Dann gibt es noch die Möglichkeit, die Zeit per Broadcast weiter zu verteilen.

```
authenticate no
```

Hier wird auf eine Authentifizierung verzichtet (aus Sicherheitsgründen keine gute Wahl, aber wenn man mit zwei »autarken« Rechnern »nur einmal testen« will, reicht das zum Probieren aus). Wenn Sie eine Konfiguration wie unten anwenden wollen, müssen Sie diese Information aus der Konfigurationsdatei löschen.

```
restrict default noserve                  # default ist deny
restrict 127.0.0.1                        # localhost darf abfragen
restrict 192.168.1.0 mask 255.255.255.0   # Hier ein klassisches Netz der
                                          # Klasse C
```

Damit werden die Maschinen festgelegt, die auf den Dienst zugreifen dürfen.

```
crypto                  # damit wird ein öffentlicher Schlüssel zugelassen
keys /usr/local/etc/ntp.keys # hier liegen die Schlüssel von NTP
trustedkey 3 4 5 14     # define trusted keys
requestkey 14           # mit diesem Schlüssel können die Einstellungen
                        # des Servers abgefragt werden
```

Konfiguration	Beschreibung
logfile /var/log/ntp	Hier werden alle Verbindungen zum NTP-Server aufgezeichnet.
server 127.127.35.1 prefer	Diese Adresse (hier eine Conrad-Uhr) ist die, die zuerst abgefragt wird, wenn es Zeit ist.
server 127.127.1.1	Das ist die virtuelle IP-Adresse für die Hardwareuhr.
broadcast 192.168.1.255	Ermöglicht das Verteilen der Zeit per Broadcast.
authenticate no	Niemand muß sich authentifizieren; wenn der Zugriff reglementiert werden soll, ist das natürlich nicht zu empfehlen.
restrict default noserve	Im Normalfall darf niemand den Server abfragen.
restrict 127.0.0.1	*localhost* darf sich die Zeit holen.
restrict 192.168.1.0 \ mask 255.255.255.0	Das Netzwerk 192.168.0 darf ebenfalls abfragen.
crypto	Ein öffentlicher Schlüssel zur Datenübertragung wird zugelassen.
keys /etc/ntp/keys	In dieser Datei finden sich die MD5-Schlüssel von ntpd.
trustedkey 3 4 5 14	Mit diesen Schlüssel und einem Passwort werden die wichtigsten Optionen von ntpd abgesichert.
requestkey 14	Hier gilt das Gleiche, allerdings nur für das Abfragen der Serverkonfiguration.
driftfile /etc/ntp/drift	Die Datei, in der definiert wird, wie weit die Systemuhr von der ermittelten Atomzeit abweichen darf.
statsdir /var/spool/ntpd	Das Verzeichnis, in dem die ntpd-Logdaten abgelegt werden.
filegen peerstats file \ peerstats type day enable	Es soll jeden Tag eine Datei angelegt werden, in der der Zustand der Peers aufgezeichnet wird.
filegen clockstats file \ peerstats type day enable	Für die Uhr wird das Gleiche getan.

Tabelle 7.6: Die wichtigsten Konfigurationsoptionen im Überblick

```
52457 43466.949 127.127.35.1 9624 -0.007356781 0.000000000 0.006915
52457 43532.943 127.127.35.1 9624 -0.009656686 0.000000000 0.006338
52457 43596.936 127.127.35.1 9624 -0.011561407 0.000000000 0.006959
52457 43662.929 127.127.35.1 9624 -0.012947057 0.000000000 0.008646
```

Abb. 7.2: Die aufgezeichneten Daten des Peers

In der ersten Spalte in Abbildung 7.2 steht, wieviel Zeit in Tagen seit dem 17. November 1858, 0 Uhr vergangen ist. Dabei handelt es sich um das sogenannte modifizierte Julianische Datum, das in der Astronomie und der Raumfahrt verwendet wird. Die zweite Spalte zeigt an, wieviel Zeit seit dem letzten Datumswechsel nach UTC in Sekunden vergangen ist. In der dritten Spalte steht die IP-Adresse des »Servers«. In der vierten Spalte ist der Zustand der Uhr in einer hexadezimalen Zahl abgelegt.

Beschrieben wird dieser Standard unter *http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/13xx/1305*.

Die fünfte Spalte zeigt an, wie falsch die Uhr geht. In der sechsten Spalte kann der Delay, die Verzögerungszeit der Synchronisation, nachgelesen werden. In Spalte sieben ist der Jitterwert der Uhr angegeben, in dem steht, wie groß die »nervösen« Schwankungen der Uhr sind.

Diese Datei kann je nach Konfiguration des Systems auch anders aussehen.

Die aufgezeichneten Daten der Uhr sehen etwa so aus wie in Abbildung 7.3.

```
52457 50570.460 127.127.35.1 2002 07 02 16 02 50
52457 50634.458 127.127.35.1 2002 07 02 16 03 54
52457 50699.454 127.127.35.1 2002 07 02 16 04 59
52457 50765.450 127.127.35.1 2002 07 02 16 06 05
```

Abb. 7.3: Die aufgezeichneten Daten der Uhr

Auch hier stehen in den ersten drei Spalten wieder das Datum, die Uhrzeit in Sekunden nach UTC-Mitternacht und die IP-Adresse des »Peers«, der Uhr. Der Rest stellt die Zeit dar, zu der die Zeile in die entsprechende Datei eingetragen wurde. Die Zeitspanne, in der neue Dateien angelegt werden sollen, können Sie Ihren Vorlieben nach auch in Monaten oder als Jahr definieren. Zu Testzwecken sollte man das aber zuerst einmal täglich tun.

Wenn Sie Ihr System konfiguriert haben, sollten Sie es testen. Das machen Sie am einfachsten mit einem Aufruf von *ntpd*. Es wird dann der in Abbildung 7.4 dargestellte Prompt angezeigt

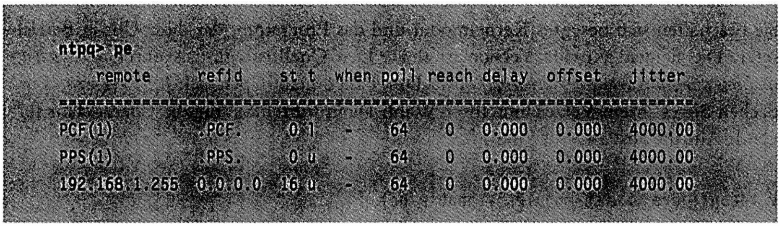


Abb. 7.4: Der Prompt des ntpd und die Ausgabe des pe-Befehls

Abbildung 7.4 zeigt die Ausgabe, die Sie erhalten, wenn die Uhr richtig erkannt wurde und der ntp gerade frisch gestartet wurde. Anschließend setzen Sie den Befehl *cv* ab. Das Ergebnis sehen Sie in Abbildung 7.5.

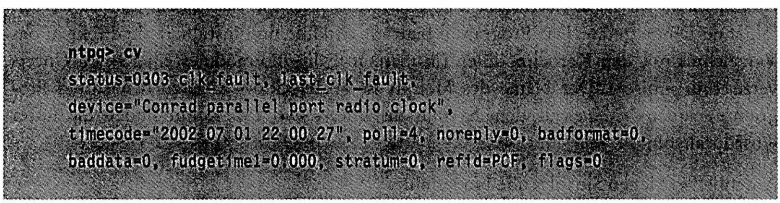


Abb. 7.5: Der cv-Befehl des ntpd

Wie in Abbildung 7.5 sehen ist, wurde die Uhr in diesem Fall korrekt erkannt. Anschließend müssen Sie sich noch eine Weile gedulden.

Wenn Sie so weit gekommen sind, sollten Sie den Rechner neu booten, damit die Uhr auch wirklich erkannt wird. Vorher sollten Sie allerdings noch ein paar anderen Operationen ausführen. In diesem Fall handelt es sich um eine der Billiguhren, die diesmal von Conrad stammt und an den Parallelport angeschlossen wird. Sie funktioniert übrigens, entgegen der Betriebsanleitung auf der beiliegenden Diskette, auch ohne Drucker ganz hervorragend.

Für diese Uhr brauchen Sie ein separates Kernelmodul, weil die Conrad-Software nur Windows unterstützt und bei Windows-Systemen die Zeit mit dieser Uhr nur beim Neustart einmal synchronisiert wird. Die Ausnahme ist, wenn die Uhr vielleicht gerade keinen Empfang hat. Dann versucht sie es nach sechs Stunden noch einmal. Wenn es dann geklappt hat, steigt die Uhr aus, es sei denn, man fordert sie manuell zur Synchronisation auf. Das ist natürlich nichts für eine genaue Zeitsynchronisation, sollte aber bei der »bekannten Stabilität« von Windows 95, für das diese Software ursprünglich bestimmt war, hinreichend sein.

Aber man hat ja neben NetBSD hoffentlich auch noch einen Linux-Rechner und kann damit Abhilfe schaffen:

Sie beschaffen sich besagtes Kernelmodul und das Programm, das diese Uhr in Betrieb setzt. Die derzeit aktuelle Version ist 0.31. Den Quelltext gibt es unter *http://www-stud.ims.uni-stuttgart.de/~voegelas/pcf.html*. Das Paket heißt »pcfclock-0.31.tar.gz«. Entpacken Sie es in ein Verzeichnis Ihrer Wahl. Dann führen Sie für die Conrad-Uhr folgende Aufrufe aus:

```
# ./configure --with-modules --enable-pcf
```

Wenn es sich um eine Maschine mit mehreren Prozessoren handelt, ist der folgende Konfigurationsaufruf durchzuführen:

```
# ./configure --with-modules  --with-smp --enable-pcf
```

Anschließend rufen Sie noch *make* und als *root* anschließend *make install* auf. Es werden folgende Dinge getan: Das Programm baut die Module und installiert sie direkt in das Verzeichnis mit den Kernelmodulen (in meinem Fall »/lib/modules/2.4.18« im Unterverzeichnis »misc«). Der Name des Moduls ist »pcfclock.o«. Zusätzlich wird ein Programm namens *pcfdate* auf dem System installiert. Das Programm finden Sie unter »/usr/local/sbin«.

Anschließend legen Sie folgende Nodes an:

```
# mknod -m 444 /dev/pcfclock c181 0
# mknod -m 444 /dev/pcfclock c181 1
# mknod -m 444 /dev/pcfclock c181 2
```

und tragen das hier in »/etc/modules.conf« ein:

```
alias char-major-181 pcfclock
```

Wenn Sie mit diesen Arbeiten fertig sind und keinen neuen Kernel bauen wollen, ändern Sie die Datei »modules.dep« in Ihrem Modulverzeichnis manuell mit dem Eintrag »/lib/modules/[Meine-Kernel-Version]/misc/pcfclock.o«.

Wenn Sie das nicht tun, wird *modprobe* nicht funktionieren und *insmod* außer der Ausgabe einer Meldung, daß die »modules.conf« neuer ist als die Module, zumindest unter RedHat ab Version 7 nichts tun.

Der Nachteil dieser Variante ist, daß Sie *pcfclock* nach jeder Neukompilation des Kernels nach diesem zugegebenermaßen etwas haarigem Verfahren neu installieren müssen.

Anschließend müssen Sie die init-Datei von ntpd nach diesem Muster ändern. Wichtig ist dabei das *pcfdate*-Kommando am Anfang der Startprozedur, das die Hardwareuhr nachstellt.

```
# Source function library.
. /etc/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -x /usr/local/bin/ntpd -a -f \
  /etc/ntp.conf ] || exit 0

RETVAL=0

start() {
  # Adjust time to make life easy for ntpd
  /usr/local/sbin/pcfdate -s
  if [ -f /etc/ntp/step-tickers ]; then
    echo -n "Syncing time for ntpd. "
    /usr/local/bin/ntpdate -s -b -p 8 \
         -u `cat /etc/ntp/step-tickers`
  fi

# Start daemons.
  echo -n "Starting ntpd: "
  daemon ntpd -A
  RETVAL=$?
  echo
  [ $RETVAL -eq 0 ] && \
    touch /var/lock/subsys/ntpd
  return $RETVAL
}

stop() {
  # Stop daemons.
  echo -n "Shutting down ntpd: "
  killproc ntpd
  RETVAL=$?
  echo
  [ $RETVAL -eq 0 ] && \
    rm -f /var/lock/subsys/ntpd
  return $RETVAL
}

# See how we were called.
case "$1" in
  start)
```

```
  start
    ;;
  stop)
    stop
    ;;
  status)
    status ntpd
    RETVAL=$?
    ;;
  restart|reload)
    stop
    start
    RETVAL=$?
    ;;
  condrestart)
   if [ -f /var/lock/subsys/ntpd ]; then
      stop
      start
      RETVAL=$?
   fi
   ;;
  *)
  echo -n "Usage: ntpd {start|stop|"
  echo "restart|condrestart|status}"
  exit 1
esac

exit $RETVAL
```

Das Programm *pcfdate* kennt drei Optionen, die in Tabelle 7.7 zusammengefaßt sind.

Aufruf	Beschreibung
pcfdate -s	Synchronisiert die Systemzeit mit der Zeit der Conrad-Uhr. Achtung: Das ist ein harter Reset der Systemzeit, der nur einmal beim Start des Systems ausgeführt werden sollte. Die anschließende Zeitsynchronisation sollten Sie dann dem ntpd überlassen.
pcfdate -u	Dieser Aufruf macht das gleiche wie *pcfdate* -s, setzt die Systemzeit allerdings auf UTC.
pcfdate -q	Damit wird die Zeit nicht an den Rechner ausgegeben.

Tabelle 7.7: Die Aufrufmöglichkeiten von »pcfdate«

Wenn Sie das System konfiguriert haben, starten Sie das System am besten neu und warten eine Weile. Wenn diese Weile vergangen ist, die in Regelfall zehn bis fünfzehn Minuten lang ist, bekommen Sie mit *ntpq* den in Abb. 7.6 gezeigten Output.

```
#ntpq> pe
     remote           refid      st t  when poll reach   delay   offset    jitter
==============================================================================
*PCF(1)          .PCF.           0 l   25   64  377    0.000   12.008    4.041
 PPS(1)          .PPS.           0 u    -   64    0    0.000    0.000 4000.00
 192.168.1.255   0.0.0.0        16 u    -   64    0    0.000    0.000 4000.00

#ntpq> cv
status=0000 clk_okay, last_clk_okay,
device="Conrad parallel port radio clock",
timecode="2002 07 02 14 49 41", poll=59, noreply=0, badformat=0,
baddata=0, fudgetime1=0.000, stratum=0, refid=PCF, flags=0
```

Abb. 7.6: Der funktionsfähige ntpd

Der »*« in Abbildung 7.6 zeigt an, welche Uhr standardmäßig benutzt wird. *cv* gibt den aktuellen Zustand der Uhr aus. In diesem Fall ist alles so weit in Ordnung. 192.168.1.255 ist die Adresse, über die die Zeit gebroadcastet wird. *offset* zeigt an, wie weit in Millisekunden die Systemuhr gerade von der externen Uhr abweicht. *when* zeigt an, zu welcher Sekunde die Abfrage durchgeführt wurde. Ab diesem Moment brauchen Sie nur noch zu warten.

Wenn der Daemon läuft, ersparen Sie sich eine Menge der eingangs genannten Probleme und können sich anderen Arbeiten als dem regelmäßigen Nachstellen der PC-Uhren widmen, was ja ab einer gewissen Anzahl von Rechnern durchaus in Arbeit ausufern kann.

7.6.4 Der ssh-Daemon

von Markus Illenseer

NetBSD installiert per Default seit einigen Versionen keinen *telnet*-Server mehr, sondern das weitaus flexiblere und sicherere *ssh*. Dabei steht die Sicherheit im Vordergrund, weswegen *ssh* auch die altbekannten r-Tools (*rsh*, *rcp*) ersetzt. Sowohl telnet als auch die r-Tools lassen sich im Netzwerk belauschen und sogar – sehr gutes Geschick vorausgesetzt – »kapern«. Passwörter werden gar unverschlüsselt im Klartext übertragen.

ssh ist genau wie die r-Tools eine Sammlung von Programmen zur sicheren Datenübertragung und zum Öffnen eines Terminals auf entfernten Servern. Um ein solches Terminal anzubieten, benötigt man auf dem Server einen Daemon für *ssh*. Dieser bedient alle *ssh*-Tools für die Authentifizierung und baut die autorisierte Verbindung auf. Die Authentifizierung geschieht bereits verschlüsselt.

Die Grundkonfiguration für ssh ist bei NetBSD ab Version 1.6 in »/etc/ssh/« zu finden. Dort befinden sich die Dateien »ssh_conf« und »sshd_conf«. Die erste Datei enthält die Konfiguration für alle *ssh*-Tools, die von diesem Server aus zu anderen Servern Kontakt aufnehmen. Die zweite Datei konfiguriert den *ssh*-Daemon.

Da der Daemon beim Systemstart mit gestartet werden muß – sonst kann man sich ja nicht remote einloggen – schauen wir uns zuerst an, wie der Daemon gestartet wird.

Dazu gibt es den Eintrag *sshd=YES* in der Datei »/etc/rc.conf«. Damit startet NetBSD beim Hochfahren des Servers die Skriptdatei »/etc/rc.d/sshd« mit der Option *start*. Dieses Skript sollten Sie auch verwenden, um den ssh-Daemon manuell zu starten oder anzuhalten. Tun Sie dies jetzt mit */etc/rc.d/sshd start*, wird der ssh-Server mit den Default-Einstellungen von NetBSD gestartet. Ein Anhalten mit der Option *stop* bewirkt nicht, daß aktuelle Verbindungen geschlossen, sondern nur, daß keine weiteren mehr angenommen werden. Um geöffnete Verbindungen zu schließen, müssen Sie mit *ps -aux* nach den *sshd*-Einträgen suchen und diese mit *kill -9* stoppen – ohne Rücksicht auf Verluste allerdings.

Eine eingehende *ssh*-Verbindung klärt mit dem Daemon die Protokoll-Version, den zu verwendenden Benutzer, dessen Authentifizierung und die zu benutzende Verschlüsselung ab. Die Protokoll-Version 1 ist obsolet, daher sollte der *ssh*-Daemon nur noch die Version 2 zulassen. Wenn Sie aktuelle *ssh*-Clients verwenden, werden Sie keine Probleme damit haben. Anders sieht es mit Clients unter anderen Betriebssystemen aus. Hier müssen Sie prüfen und testen. Der Eintrag *Protocol 2,1* in der Konfigurationsdatei »/etc/ssh/sshd_conf« enthält die erlaubten Protokolle in der zu favorisierenden Reihenfolge. Tragen Sie hier nur die 2 ein, um das obsolete Protokoll 1 zu vermeiden.

Eine weitere wichtige strategische Entscheidung ist, ob Sie zulassen wollen, ob der Benutzer root sich von einem anderem Rechner per *ssh* verbinden darf. Die Faulheit und Bequemlichkeit hin oder her, das Risiko ist hoch, daß hier versehentlich oder absichtlich etwas schief geht. Hackers Tür ist geöffnet. Das Verbot, daß root sich direkt einloggen kann, wird mit dem Eintrag *PermitRootLogin* gesetzt, der auf *no* stehen muß. Loggen Sie sich nun als ein normaler Benutzer ein, können Sie mit dem Befehl *su* die Funktion des Benutzers *root* übernehmen.

Alternativ zum Standard TCP-Port 23 kann man den *ssh*-Daemon auch auf beliebigen anderen Ports horchen lassen, auch von unterschiedlichen IP-Nummern aus. Wenn Ihr Rechner zum Beispiel als Server oder als Gateway fungiert, ist das eine interessante Lösung. Wirklich sinnvoll ist es allenfalls, Port 22 durch Port 443 zu ersetzen, da dieser Port in vielen Firmen-Firewalls freigeschaltet ist. Aber der Port ist zumeist schon für den Apache und ssl reserviert. Daher die Idee mit der weiteren IP-Nummer.

Tragen Sie dies in der Konfiguration ein, lauscht der *ssh*-Daemon auf zwei IP-Nummern mit jeweils einem Port:

```
ListenAddress 0.0.0.0
ListenAddress 192.168.0.47:443
ListenAddress ::
```

Der erste Eintrag steht für »Lausche auf allen eingestellten Default-IP-Nummern und dem Standard-Port«. Der zweite Eintrag steht für »Lausche auf der IP-Nummer 192.168.0.47 mit dem Port 443«. Der dritte Eintrag beendet den Konfigurationsblock. Es ist erforderlich, daß die zweite IP-Nummer im System eingetragen wird. Siehe auch *ifconfig*.

Nach jeder Änderung in der Konfiguration müssen Sie dem Daemon sagen, daß er diese neu laden soll. Das geschieht mit der Option *reload* für das Start- und Stop-Skript.

Zugangskontrolle

Der Zugang per *ssh* ist nun für jedermann gestattet, der einen Account besitzt und sein Passwort kennt. Dies kann durch Regeln beeinflußt werden, die zumeist IP-basierte Netzwerke sperren beziehungsweise erlauben. Wenn Sie Ihren Server für die weite Welt sperren möchten, etwa weil Sie per xDSL stets im Internet arbeiten, wird das mit einer Erweiterung der Datei »/etc/hosts.allow« festgelegt. In der Datei »/etc/ssh/sshd_conf« kann man mit der Option *AllowUsers* einzelne Benutzer aussperren oder erlauben:

```
AllowUsers joe
```

Nun darf sich nur der Benutzer *joe* anmelden, niemand sonst. Weitere Informationen gibt es in der Manpage *sshd_config* und *hosts.allow*.

7.6.5 Yellow Pages

von Dr. Lex Wennmacher

Die Yellow Pages (YP oder Network Information System, NIS) stellen ein Gerüst zur Verteilung von Informationen wie beispielsweise Paßwörter, Gruppenzugehörigkeit und Mail-Aliase in einem Netzwerk zur Verfügung. Wer zum Beispiel dafür sorgen will, daß seine Benutzer sich mit dem gleichen Accountnamen und Passwort auf mehreren Rechnern einloggen können, kann dafür die Yellow Pages einsetzen. Folgende Maps können mit den Yellow Pages netzwerkweit zur Verfügung gestellt werden: *passwd*, *group*, *aliases* (Mail-Aliase), *ethers*, *hosts*, *netgroup* (Gruppen von Hosts), *networks*, *protocols*, *rpc*, *services*, *netid* und *amd.home*.

»Yellow Pages« (abgekürzt YP) und »Network Information System« (abgekürzt NIS) sind synonmym. Wie viele andere Netzwerk-Dienste wurde YP von der Firma Sun Microsystems entwickelt und auf den Markt gebracht. Sun wählte den Namen »Yellow Pages« in Anlehnung an die von der Post in vielen Ländern herausgegebenen »Gelben Seiten«, da die gebotene Katalogfunktion durchaus vergleichbar ist. Allerdings hatte Sun nicht beachtet, daß die Rechte am Namen »Yellow Pages« unter anderem in den USA, Kanada und Großbritannien von der jeweiligen Post bereits gesichert waren. Deswegen kam es 1989 zu einem Prozeß, den Sun verlor. Als Konsequenz benannte Sun 1990 YP in NIS um. Alle YP-Kommandos beginnen mit »yp...« und interessanterweise wurde (aus Kompatibilitätsgrunden) dies beibehalten.

Eine weitere Vorbemerkung ist angebracht: Das Standard-YP ist ein unsicherer Dienst. Die verschlüsselten Passworte aus dem *pw_passwd*-Feld werden im Klartext über das Netzwerk transportiert, jeder unprivilegierte Benutzer kann sich mit dem Befehl

```
% ypcat passwd
```

alle über YP verteilten Accounts und (verschlüsselte) Passworte anzeigen lassen. Einem anschließenden Versuch, Accounts mit Hilfe von *crack* zu knacken, steht nichts mehr im Weg. Moderne Unix-Systeme beugen dieser Art von Angriffen dadurch vor, daß das *pw_passwd*-Feld nur in der vom Normalbenutzer nicht lesbaren Datei »/etc/master.passwd« gehalten wird.

Die Unix-BSD-Familie entwickelte eine Lösung dieses Sicherheitsproblems, die aber nicht kompatibel ist und bei der Teilnahme von Nicht-BSD-Systemen am YP-System daher nicht funktioniert. Diese Lösung sieht vor, daß ein Eintrag in der Datenbasis den YP Server *ypserv* (siehe unten) darüber informiert, daß die Map sicherheitsrelevante Daten enthält, die nicht an Clients weitergereicht werden, die nicht an reservierte Ports gebunden sind. Solche Maps kann man durch Angabe der Option *-s* des Befehls *makedbm* erzeugen.

Dennoch sollte man bedenken, daß auch bei dieser Lösung die verschlüsselten Passworte im Klartext über das Netzwerk übertragen werden. In einem nach außen (zum Internet hin) von einem Firewall geschützten Netzwerk, das zudem noch geswitcht sein sollte, ist normalerweise die gebotene Sicherheit dieser Lösung in Ordnung. Wer mehr Sicherheit benötigt, darf YP nicht benutzen und sollte sich statt dessen (zumindest für die Benutzer-Authentifikation) beispielsweise den Kerberos-Dienst ansehen.

Die Komponenten

Ohne ins Detail gehen zu wollen, hier ein Überblick über die Daemonen und wichtigsten Befehle des YP-Systems:

Ypserv	Der YP-Server-Daemon, läuft auf allen YP-Servern und beantwortet via RPC eintreffende YP-Anfragen.
Ypbind	Der YP-Client-Daemon, läuft auf allen YP-Clients, bindet sich an einen YP-Server und stellt Anfragen an ihn via RPC.
rpc.yppasswdd	Der Passwort-Änderungs-Daemon, läuft auf dem YP-Master-Server (siehe unten) und sorgt dafür, daß Benutzer auch ihr via YP verteiltes Passwort ändern können.
Ypinit	Shellskript zur Konfiguration von YP.
ypcat	Dienstprogramm zum Abfragen von YP-Maps.
ypwhich	Dienstprogramm zum Bestimmen des YP-Servers, an den der YP-Client gerade angebunden ist.

Die zentrale Rolle im YP-System übernimmt der Master-Server, auf dem die zu verteilenden Datenbasen als Quelldateien zentral liegen. Auf dem Master-Server läuft der *ypserv*, der die Datenbasen ins Netz propagiert. Ausfallsicherheit wird erreicht, indem man bis zu drei sogenannte Slave-Server definiert. Die Slave-Server unterscheiden sich vom Master-Server darin, daß sie ihre Datenbasen nicht aus Quelldateien ableiten, sondern ihre Maps direkt vom Master-Server erhalten. Zum Verteilen der Datenbasen läuft auf ihnen ebenfalls der *ypserv*-Prozeß. Alle anderen Knoten im YP-System sind sogenannte Clients. Auf ihnen läuft zur Nutzung des YP-Diensts der *ypbind*-Prozeß.

YP setzt, genau wie NFS, auf den Remote-Procedure-Call-Diensten (RPC) auf (für eine Beschreibung von RPC siehe Kapitel 9.1). Es ist daher erforderlich, daß der *rpcbind*-Daemon auf allen YP-Maschinen läuft.

Im folgenden wird das Einrichten eines Master-Servers, eines Slave-Servers und eines Clients beschrieben. Wer ein YP-System von Grund auf einrichten will, muß zuerst den Master-Server, dann die Slave-Server und zum Schluß die Clients einrichten. Um ein bestehendes YP-System auszubauen, kann man einen NetBSD-Rechner entweder als Slave-Server hinzufügen oder man setzt NetBSD-Rechner lediglich als Clients ein.

Domainname

Unabhängig davon, welche Sorte von Server oder Client man einrichten will, muß man als erstes den YP-Domain-Namen setzen. Eine YP-Domain bezeichnet die Gesamtheit aller Rechner, die gemeinsame YP-Maps benutzen. In einer YP-Domain gibt es genau einen Master-Server. Jeder Rechner kann nur maximal einer YP-Domain angehören.

Mit dem Befehl

```
% domainname
```

fragt man den aktuell gesetzten YP-Domain-Namen ab und *root* kann den Domain-Namen mit

```
# domainname <YP-Domain-Name>
```

setzen. Man beachte, daß dieser Domain-Name und der *domainname*-Befehl nichts mit dem voll qualifizierten Hostnamen (à la *host.domain.net*) zu tun hat.

Üblicherweise definiert man den Domain-Namen in der Datei »/etc/rc.conf« mit Hilfe eines Eintrags

```
domainname="beispieldomainnamen"
```

Im Verlauf des Boot-Vorgangs setzt dann »/etc/rc.d/network« den Domain-Namen.

Master-Server konfigurieren

Wer eine neue YP-Domain einrichten will, sollte als erstes einen YP-Domain-Namen wählen, den Namen in »/etc/rc.conf« eintragen und den Domain-Namen, wie vorhin beschrieben, mit dem *domainname*-Befehl setzen. Als Domain-Namen sind Zeichenketten mit einer Länge von bis zu *MAXHOSTNAMELEN* (256 in NetBSD 1.6) Zeichen zulässig. Da der Domain-Namen als Directory-Name verwendet wird, wird empfohlen, keine Zeichen zu nehmen, die eine Sonderbedeutung für die Shell haben, und, falls möglich, auch keine Leerzeichen. Falls der einzurichtende Master-Serer eine bestehende YP-Domain bedienen soll, trägt man den Namen dieser Domain ein.

Ein weiteres Wort vorweg: die hier beschriebene Installation weicht vom üblichen Verfahren, einen Master-Server unter NetBSD einzurichten, ab. Üblicherweise geht dies mit dem Befehl

```
# ypinit -m <master server> <domain name>
```

Er sorgt dafür, daß die lokalen Dateien des Master Servers, also zum Beispiel »/etc/master.passwd«, »/etc/group«, »/etc/services« und so weiter als Quelldateien verwendet werden. Dieses Vorgehen ist in mehrfacher Weise problematisch. Zunächst bedeutet es, daß alle Benutzer, deren Passwort-Information man via YP verteilen will, auf dem Master-Server einen Account besitzen müssen. Oft will man jedoch auf zentralen Server-Maschinen keine Benutzer zulassen. Ein weiterer Punkt ist, daß nun auch beispielsweise das Root-Passwort des Master-Servers über YP verbreitet wird.

Aus diesen Gründen wird hier eine wesentlich flexiblere alternative Konfiguration beschrieben, die nur geringfügig höheren Aufwand nach sich zieht.

```
# cd /var/yp
# mkdir `domainname`
# cd `domainname`
# mkdir etc
```

In das Verzeichnis »/var/yp/`domainname`« kommen die zu verteilenden Datenbasen mit Ausnahme von »master.passwd« zu liegen, also zum Beispiel »group«, »mail.aliases« und so weiter. Es empfiehlt sich, diese Dateien mit Hilfe eines Revison Control Systems (am besten RCS, siehe *man rcsintro*) zu verwalten:

```
# mkdir RCS
# vi group
# ci -l group
# vi master.aliases
# ci -l master.aliases
```

... und so weiter, für alle Datenbasen. Als nächstes erzeugt man sich eine geeignete Datei »Makefile«:

```
# cp ../Makefile.yp Makefile
```

In dieser Makedatei muß man nun einige Anpassungen vornehmen.

1. Falls man Nicht-BSD Rechner in der YP-Domain hat, muß man

   ```
   INSECURE?=      yes
   ```

   setzen.

2. Sowohl der Eintrag für *DIR* als auch für *AMDDIR* muß geändert werden:

   ```
   DIR=        /var/yp/beispieldomainnamen
   AMDDIR=     /var/yp/beispieldomainnamen
   ```

3. Es brauchen nur diejenigen Maps neu erzeugt zu werden, die man auch wirklich via YP verteilen will. Daher ändert man den Eintrag für *all:* im Makefile entsprechend ab. Beispiel:

   ```
   all:: passwd aliases group
   ```

   Die Datenbasen für *passwd*, *aliases* und *group* werden über YP verteilt. Der doppelte Doppelpunkt ist empfohlen.

4. Außerdem nimmt man noch folgende Änderungen vor:

   ```
   passwd.time: ${DIR}/master.passwd
   ```

   ersetzen durch:

   ```
   passwd.time: ${YPDBDIR}/<domainnamen>/etc/master.passwd
   ```

   ```
   aliases.time: ${DIR}/mail/aliases
   ```

   ersetzen durch:

   ```
   aliases.time: ${DIR}/mail.aliases
   ```

   Am Ende des Makefiles wird:

   ```
   ${DIR}/mail/aliases:
   ```

   durch

   ```
   ${DIR}/mail.aliases:
   ```

   ersetzt und

   ```
   ${DIR}/master.passwd:
   ```

   durch

   ```
   ${DIR}/etc/master.passwd:
   ```

Mit Hilfe der so geänderten Datei »Makefile« erzeugt man sich (aus den Quelltext-Dateien »group«, »mail.aliases« und so weiter) die via YP verteilten Maps:

```
# make group
# make aliases
```

und so weiter. Eine Ausnahme stellt die »master.passwd« dar. Damit Benutzer ihr YP-Passwort ändern können, muß auf dem YP-Master-Server der Daemon *rpc.yppasswdd* laufen. Er ändert, wenn er ohne weitere Optionen aufgerufen wird, bei Bedarf die Paßwörter in der »/etc/master.passwd« – was wir ja, wie oben beschrieben, nicht wollen. Es gibt aber die Möglichkeit, dem *rpc.yppasswdd* ein Directory-Präfix anzugeben, relativ zu dem der Daemon dann »etc/master.passwd« sucht. Da wir diese Option benutzen wollen, wählen wir also als Ort für die Passwort-Datei »/var/yp/<domainnamen>/etc/master.passwd«. Nun können wir die Passwort-Map erzeugen:

```
# make master.passwd
```

Nach jeder Änderung, die man in den Maps auf dem YP-Master-Server vornimmt, werden die geänderten Maps zu den YP-Slave-Servers »gepusht« (dieser Mechanismus ist im eben editierten Makefile enthalten – es sei den, man setzt die Variable *NOPUSH* auf *No*). Der Master-Server muß also die Slave-Server kennen, damit er weiß, wem er die Maps pushen soll. Dafür gibt es eine eigene YP-Map, in der die Slave-Server verzeichnet sind, sie muß bei der Konfiguration des Master-Servers angelegt werden. Dazu trägt man in der Datei mit dem Namen »ypservers« die Namen der Slave-Server ein, und zwar wird für jeden Slave-Server einen Eintrag erzeugt, der aus dem voll qualifizieren Namen des Slave-Servers, gefolgt von einem Leerzeichen, gefolgt von der Wiederholung des voll qualifizieren Namens, besteht. Beispiel:

```
# $Id: kap7.txt,v 1.10 2002/12/09 10:12:44 wennmach Exp wennmach $
slaveserver1.domain.net slaveserver1.domain.net
slaveserver2.domain.net slaveserver2.domain.net
slaveserver3.domain.net slaveserver3.domain.net
```

Die entsprechenden YP Maps, die der Master-Server benutzt, werden so angelegt:

```
# make ypservers
updated ypservers
```

Soll der Master-Server selbst Yellow Pages einsetzen (was in der Regel der Fall sein dürfte), muß auf ihm auch der YP-Client *ypbind* laufen. Falls *ypbind* nicht mit der Option *-broadcast* gestartet wird (von dieser Option ist aus Sicherheitsgründen abzuraten), bestimmt es seine YP-Server aus den Einträgen in der Datei »/var/yp/binding/<domain>.ypservers«. Sie erzeugt man beispielsweise so:

```
# hostname > ../binding/`domainname`.ypservers
# awk '{print $1}' ypservers >> ../binding/`domainname`.ypservers
```

Bevor man nun den *ypserv* startet, überzeugt man sich davon, daß alle zu verteilenden Datenbasen vorhanden sind:

```
# ls *.db
group.bygid.db          mail.byaddr.db          passwd.byname.db
group.byname.db         master.passwd.byname.db passwd.byuid.db
mail.aliases.db         master.passwd.byuid.db  ypservers.db
```

Nun bleibt noch, die notwendigen Einträge in »/etc/rc.conf« vorzunehmen und die Dienste zu starten:

```
rpcbind=YES
rpcbind_flags="-l"
ypbind=YES
ypserv=YES
yppasswdd=YES
yppasswdd_flags="-d /var/yp/beispieldomainnamen"

# /etc/rc.d/rpcbind start
# /etc/rc.d/ypserv start
# /etc/rc.d/yppasswdd start
# /etc/rc.d/ypbind start
```

Slave-Server konfigurieren

Auch die Konfiguration des Slave-Servers beginnt mit dem Setzen des YP-Domainnamens in der »/etc/rc.conf«, unter Umständen gefolgt von einem manuellen Aufruf von

```
# domainname <beispieldomainnamen>
```

Ebenfalls in der »/etc/rc.conf« wird vermerkt, daß der *rpcbind*-Dienst sowie der *ypserv* beim Booten gestartet werden sollen:

```
rpcbind=YES
rpcbind_flags="-l"
ypbind=YES
ypserv=YES
```

Den Rest der Konfiguration erledigt nun aber *ypinit*. Man muß lediglich die Option *-s* (zum Zeichen, daß man einen Slave-Server konfiguriert) gefolgt vom Namen des Master-Servers, sowie optional den Namen der YP-Domain angeben. Beispiel:

```
# ypinit -s masterserver.domain.net beispieldomainnamen
```

Jetzt werden noch die YP-Dienste gestartet:

```
# /etc/rc.d/rpcbind start
# /etc/rc.d/ypserv start
# /etc/rc.d/ypbind start
```

...und fertig!

Client konfigurieren

Auch hier beginnen wir mit der Definition der YP-Domain in »/etc/rc.conf« (gegebenenfalls gefolgt von einem händischen Setzen des Domain-Namens mit *domainname*) und mit dem Anwählen der Dienste *rpcbind* und *ypbind*:

```
rpcbind=YES
rpcbind_flags="-l"
ypbind=YES
```

Wiederum erledigt *ypinit* den Rest der Konfiguration:

```
# ypinit -c

Server type: client
Domain:      beispieldomain

A YP client needs a list of YP servers to bind to.
Whilst ypbind supports -broadcast, its use is not recommended.

Please enter a list of YP servers, in order of preference.
When finished, press RETURN on a blank line or enter EOF.

        next host: masterserver.domain.net
        next host: slaveserver1.domain.net
        next host: slaveserver2.domain.net
        next host: slaveserver3.domain.net
        next host:

The current servers are:
```

```
masterserver.domain.net
slaveserver1.domain.net
slaveserver2.domain.net
slaveserver3.domain.net

Is this correct? [y/n: n] y
#
```

Die Angabe des YP-Domainnamens ist nicht notwendig, wenn er vorher mit Hilfe von *domainname* gesetzt wurde.

Pflege der YP-Dateien

Wie bereits erwähnt, sollte man die YP-Quelldateien auf dem Master-Server mit Hilfe von RCS pflegen. Spezielles Augenmerk verlangen manuelle Eingriffe in »master.passwd«. Hier muß man verhindern, daß die manuellen Änderungen mit Änderungen durch den *rpc.yppasswdd* interferieren. Das wird durch folgende Prozedur sichergestellt:

```
# cd /var/yp/beispieldomain/etc
# cp master.passwd ptmp
# vi ptmp
# pwd_mkdb -d /var/yp/beispieldomain ptmp
# ci -l master.passwd
# cd ..
# make
```

Die Existenz der Datei »ptmp« dient als Sperre gegen Änderungen durch den *rpc.yppasswdd*. Das abgesicherte Zurückkopieren übernimmt das Dienstprogramm *pwd_mkdb*, das auch die Datei »passwd« erstellt (neben den nicht benötigten Datenbasen »pwd.db« und »spwd.db«).

7.6.6 Sendmail

Von Thorsten Stritzinger

Sendmail ist der berühmteste und am weitesten verbreitete Nachrichtenübertragungsagent (Mail Transfer Agent – MTA) in der BSD-Welt und mit Sicherheit nicht nur dort. Im Gegensatz zur öffentlichen Meinung ist Sendmail gewiß nicht so schlecht wie behauptet, weder was die Sicherheit angeht, noch was das Meistern der Syntax von »sendmail.cf«, der mächtigsten Konfigurationsdatei von Sendmail, betrifft. Ehrlicherweise muß man zugeben, daß die Syntax der »sendmail.cf« sehr fragwürdig ist und auch nur von einigen Hardcore-Sendmail-Gurus richtig beherrscht wird. Im folgenden Abschnitt wird unter anderem demonstriert, daß man keine Angst vor Sendmail haben muß, da sich auch damit schnell ein funktionierendes und sicheres SMTP-Gateway einrichten läßt.

Das Licht der Weilt erblickte Sendmail im Jahr 1983 mit dem 4.1cBSD, seit diesem Zeitpunkt gibt es keine Anwendung seiner Art, der mehr vertraut wurde.

Unter den MTAs ist Sendmail das große Arbeitstier des Internets, es überträgt verläßlich E-Mails zwischen den Netzen. Nicht ohne Grund wird er standardmäßig bei NetBSD installiert. Aber wie jedes Programm besitzt Sendmail sowohl Vor- als auch Nachteile.

Auf der positiven Seite weist es eine sehr große Gemeinschaft an Benutzern auf, was zur Folge hat, daß man schnell und ohne Probleme sowohl freie als auch kommerzielle Unterstützung findet. Ein weiterer Punkt ist die Fülle an Publikationen für Sendmail, die ihresgleichen sucht. Aber auch der inzwischen erlangte Reifegrad dieser Software, der dafür verantwortlich ist, daß Sendmail so stabil und voraussagbar arbeitet, ist ein deutlicher Pluspunkt.

Auf der negativen Seite enthält Sendmail aufgrund seiner langen Entwicklungszeit eine Menge veralteten Codes, worauf sich der Ruf zurückführen läßt, unsicher zu sein und immer aufgeschwemmter zu werden. Über diese Behauptungen kann man natürlich diskutieren. Wahr ist jedoch mit Sicherheit, daß es über die Jahre eine große Anzahl Angriffspunkte für Hacker (Vulnerabilities) gegeben hat, allerdings wurden diese schnell entdeckt und ausgebessert. Sendmail ist vom Code her wesentlich größer ist als andere MTAs wie QMail von Dan Bernstein und Postfix, auch der RAM-Verbrauch ist wesentlich größer. Das hat allerdings mehr mit der Tatsache zu tun, daß Sendmail monolithisch aufgebaut ist, also ein einziges Programm die ganze Funktionalität bietet. Man könnte auch sagen, daß Sendmail quasi aus einem Guß ist. De facto wurde der Quellcode von so vielen Programmierern über die Jahre analysiert, gebugfixed und optimiert, daß viel veralteter Code über die letzten zwanzig Jahre gar nicht unbeschadet überlebt haben kann. Es ist wesentlich sinnvoller und anschaulicher, Sendmail als monolithischen Block anzusehen, weshalb Sendmail auch mit root-Rechten laufen muß, sobald irgendeine erforderte Funktion root-Rechte benötigt, wie zum Beispiel das Schreiben von Nachrichten an mehrere Home-Verzeichnisse.

Ein anderer Kritikpunkt ist seine Komplexität. Die Syntax der Haupt-Konfigurationsdatei »sendmail.cf« ist alles andere als intuitiv. Dieser Punkt ist, wie wir später noch sehen werden, nicht mehr ganz so richtig, da es seit einiger Zeit möglich ist, die Konfiguration über sogenannte m4-Makros zu durchzuführen, was um ein vielfaches benutzerfreundlicher ist als das direkte Editieren der Datei »sendmail.cf«.

Ungeachtet aller verschiedenen Meinungen und Standpunkte zu Sendmail handelt es sich unbestreitbar um ein mächtiges und gut unterstütztes Softwarepaket. Der Nutzen von Sendmail überwiegt für sehr viele die negativen Seiten.

Die Architektur von Sendmail

Die Kontrollinstanz bei Sendmail schlechthin ist die »sendmail.cf«. Diese Datei wird mit dem Makroprozessor *m4* aus der Datei »sendmail.mc« generiert, man kann allerdings die »sendmail.cf« auch direkt bearbeiten. Sie schwebt quasi über allen anderen Konfigurations- und Sendmail-Datenbank-Dateien, da sie angibt, ob diese Dateien überhaupt befragt oder verwendet werden. Für die meisten Zwecke kann das von der Distribution

mitinstallierte »sendmail.cf« übernommen werden, wer jedoch Sendmail richtig nutzen möchte, muß (wenigstens) kleinere Modifikationen vornehmen.

Im Prinzip besteht die »sendmail.cf« aus zwei Teilen: Der erste Teil ist gut kommentiert und legt wichtige Sachen wie Konfigurationsdateien, Format, Hostnames und Loglevel (dies gibt die Genauigkeit der Log-Datei an) fest. Diesen Teil muß man kaum editieren. Der zweite Teil ist für den schlechten Ruf bezüglich der Konfigurierbarkeit von Sendmail verantwortlich, da er wirklich sehr kryptisch ist. Andererseits bekommt man hier sofort einen Eindruck von der Mächtigkeit, die Sendmail innewohnt.

Verzeichnisse und Konfigurationsdateien

Bei der Installation von Sendmail werden einige wichtige Verzeichnisse und Dateien angelegt:

In »/var/spool/mail« werden alle ankommenden E-Mails im mbox-Format gespeichert. Das bedeutet, daß alle E-Mails für einen Benutzer nach ihrem zeitlichen Eintreffen aneinander gehängt werden.

Das Verzeichnis »/var/spool/mqueue« bildet eine sogenannte E-Mail-Warteschlange (Mail Queue) für die ausgehende elektronische Post, die nicht sofort zustellbar ist.

Das Verzeichnis »/etc/mail« enthält alle Dateien für die Konfiguration und unter Umständen auch ein Makefile, das aus den Konfigurationsdateien die zugeordneten *.db-Dateien erzeugt. Ganz wichtig ist, daß nur diese *.db-Dateien von Sendmail verwendet werden. Das hat natürlich zur Folge, daß sämtliche Änderungen in den Konfigurationsdateien ohne Effekt bleiben, wenn danach nicht das beziehngsweise die entsprechenden *.db-Dateien neu erzeugt werden. Dies ist übrigens ein häufiger Fehler von Sendmail-Neulingen. Ist kein Makefile zu finden, muß man die *.db-Dateien von Hand generieren, was wie folgt gemacht wird :

```
cat ./<Dateiname> | makemap hash <Dateiname>
```

Aber wie auch anders gibt es hier wieder eine Ausnahme zur Regel und zwar handelt es sich um die Datei »aliases.db«, die als einzige mit dem Kommando *newaliases* generiert wird. Vorsicht: Hier lauert wieder eine kleine Falle für Anfänger, denn mit Hilfe des Befehls *makemap* entsteht auch eine neue »aliases.db« und zwar zugegebenermaßen eine sehr giftige »aliases.db«, die sogar in der Lage ist, Sendmail komplett funktionsunfähig zu machen.

Anmerkung:

Bei Versionen aber, als NetBSD 1.5 befinden sich die Konfigurationsdateien in »/etc« anstatt wie hier in »/etc/mail«.

Die wichtigsten Konfigurationsdateien neben der »sendmail.cf« werden nachfolgend kurz beschrieben:

access

Die »/etc/mail/access« beziehungsweise die daraus entstandene Datei »access.db« bestimmt jedes Mal, wenn versucht wird, eine E-Mail zum Versenden oder Empfangen an Sendmail zu übergeben, ob sich Sendmail wirklich dieser Arbeit annehmen oder sie verweigern soll. Ein einfaches Beispiel ist in Listing 7.1 dargestellt.

```
# Aufbau der /etc/mail/access
# HOST                    POLICY

# Standard-Eintraege
localhost                 RELAY
127.0.0.1                 RELAY

# eigene Eintraege
mydomain.com              RELAY
fallback.com              RELAY
```

Listing 7.1: Eine einfach mail/access-Datei

local-host-names und virtusertable

In Abbildung 7.7 ist ein einfacher Vorgang schematisch abgebildet, der nun kurz erläutert wird:

Eine für die Adresse *thorsten@mydomain.com* eingehende E-Mail soll akzeptiert und in die entsprechende mbox geschrieben werden. Für diesen Zweck müssen zwei weitere Konfigurationsdateien vorhanden sein, nämlich »local-host-names«[1] und »virtusertable«. Erstere Datei ist im Prinzip nichts anderes als eine Liste mit Domains, die als lokal betrachtet werden sollen. In unserem konkreten Beispiel gibt es nur einen Eintrag, nämlich *mydomain.com*. Eine solche Einstellung ist sinnvoll, da beispielsweise bei einem ISP ein Mailhub im Normalfall für sehr viele unterschiedliche Domains verantwortlich ist, was wiederum bedeutet, daß er sie alle als lokal betrachten muß und zwar selbst dann, wenn er zu einer ganz anderen Domain gehört.

Es gab einige Namens- und Pfadänderungen, zum Beispiel änderte sich der »sendmail.st« in »statistics«. Lesen Sie hierzu mehr in der »README.cf«.

An Listing 7.2 kann man den prinzipiellen Aufbau von »virtusertable« sehen. Wichtig dabei sind drei unterschiedliche Fälle. Die E-Mails an *thorsten@mydomain.com* werden in die mbox »thormy« gespeichert. Die E-Mails für *little.sister@mydomain.com* werden wei-

[1] Die Datei »local-host-names« ist in älteren Sendmail-Versionen als »sendmail.cw« bekannt.

tergeleitet an *little.sister@daheim.de* und alle weiteren E-Mails für die Domain *mydomain.com*, egal ob *contact@mydomain.com*, *yoda@mydomain.com* oder wie auch immer, werden auch in die mbox »thormy« abgelegt.

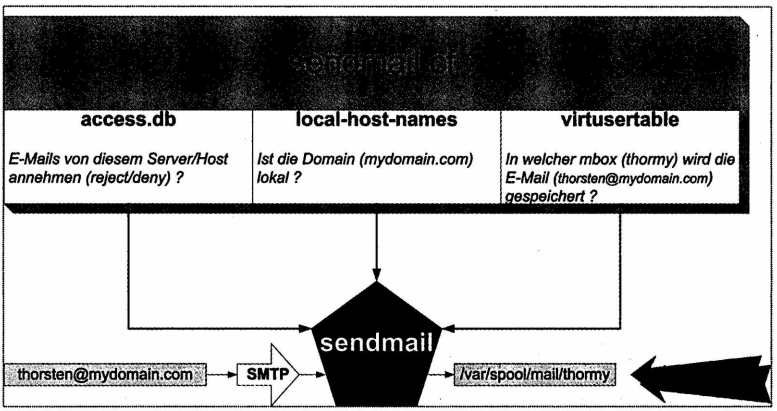


Abb. 7.7: Eingehende Nachricht

Abbildung 7.8 stellt den entgegengesetzten Fall dar. *onehost.mydomain.com* hat eine E-Mail für *ziel.de* geschrieben und möchte, daß sie von Sendmail zugestellt wird, was mit den besprochenen Einstellung problemlos funktionieren sollte.

```
# Beispiel-Konfiguration von /etc/mail/virtusertable

thorsten@mydomain.com                   thormy
little.sister@mydomain.com              little.sister@daheim.de

# @mydomain.com                         %1@yourdomain.com
# hiermit werden alle E-Mails für mydomain.com an yourdomain.com
# weitergeleitet, der Name vor dem @ bleibt dabei erhalten
```

Listing 7.2: Eine Beispiel-Konfiguration von »/etc/mail/virtusertable«

mailertable

Die Spezifikation der Domain-Name-Server (DNS) definiert für besondere Zwecke des Mail-Routings Mail Exchanger Resource Records, kurz MX-Records. MB, MG, MINFO und MR Resource Records sind alternative Konzepte, die sich jedoch nicht durchsetzen konnten. Ein MX-Record spezifiziert einen Domain-Namen und einen zugehörigen Rechner, den Mail-Exchanger, der in der Lage ist, Mails an eine E-Mail-Adresse mit

diesem Domainnamen auszuliefern. Der Domain-Teil einer E-Mail-Adresse braucht kein Domainname eines existierenden Rechners zu sein. Es können mehrere Mail Exchanger für den gleichen Domainnamen angegeben werden. Mailsysteme können so konfiguriert werden, daß sie einen Name-Server nach MX-Einträgen für den Domain-Teil einer gegebenen Zieladresse abfragen. In der Praxis macht man im Domain-Name-Server (DNS) neben dem Haupt-Mail-Server einer Domain noch ein zweiten MX-Record Eintrag geringerer Priorität, dieser zweite Eintrag definiert einen sogenannten Fallback-Server, damit man wenigstens einen Ersatzserver in der Hinterhand hat. Tritt nun der Fall ein, daß der Haupt-Mailserver, also der Mailserver mit der höchsten Priorität, nicht zu erreichen ist, dann gehen alle E-Mails an den Mailserver mit der nächsthöheren Priorität, wobei dieser nicht in der gleichen Domain liegen muß. Abbildung 7.9 zeigt diese Konfiguration.

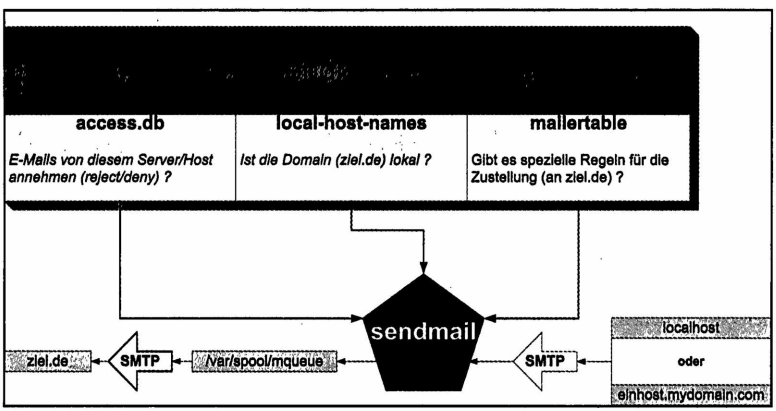


Abb. 7.8: Ausgehende Nachricht

fallback.com ist bereits in der »access.db«, also werden auch die für *fallback.com* eingehenden E-Mails von Sendmail weiterverarbeitet. Die Datei »local-host-names« enthält nur *mydomain.com* als Eintrag, aus diesem Grund wird *fallback.com* nicht als lokal angesehen. An diesem Punkt tritt die Datei »mailertable« in Aktion. Sie besitzt den Eintrag

```
fallback.com smtp:[IP-Adresse von fallback.com]
```

Diese Angabe ist unter Umständen redundant, was jedoch keine negativen Folgen zur Konsequenz hat. Wenn die eckigen Klammern gesetzt sind, ist es möglich, hier alternativ die IP-Adresse jedes anderen erreichbaren Mail-Hubs einzutragen. Die MX-Records bezüglich *fallback.com* können vollkommen vernachlässigt werden. Die für *fallback.com* ankommenden E-Mails werden nun in eine Warteschlange (Queue) gestellt, Sendmail versucht nun, diese in gewissen Abständen auszuliefern. Nach fünf Tagen wird die Queue geleert, egal, ob Sendmail bei der Zustellung erfolgreich war oder nicht.

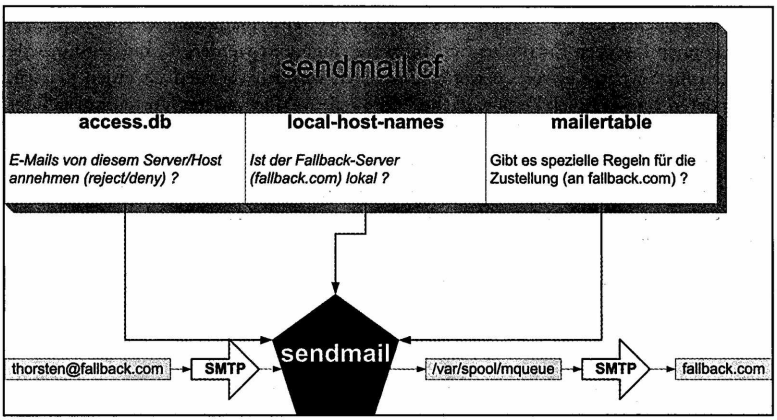


Abb. 7.9: Konfiguration für einen Fallback-Server

aliases

Äußerst spannend wird »mailertable« erst im Zusammenspiel mit der »aliases.db«, da man damit die für *mydomain.com* ankommenden E-Mails an verschiedene Standorte beziehungsweise Mailhubs verteilen kann, je nachdem, wer als Empfänger vor dem »@« steht. Nun kommen auch die eckigen Klammern, die den Domain-Name-Server (DNS) außer Kraft setzen, voll zur Entfaltung. Die »aliases.db« ist meiner Meinung nach eines der mächtigsten Features von Sendmail im professionellen Betrieb, da sie bestimmt, ob die Empfängeradresse, die Bestandteil der als lokal betrachteten Domain(s) sein muß (»local-host-names«), in eine andere Adresse umgewandelt werden soll oder nicht. Die neue Adresse könnte beispielweise ein in-house Exchange-Server, die Adresse einer Firmen-Filiale an einem anderen Ort oder eine Mailing-Liste sein.

```
# Beispielkonfiguration von /etc/mail/aliases
# Empfaengeradresse          Adresse mit der der Empfaenger
#                            maskiert werden soll.
thorsten:                    thorsten@exchange.mydomain.com

# oder auch
sister:                      little.sister@berlin.mydomain.com

# lokale mbox(en), POP3
abt_1:                       karl,detlef,sabine
```

Listing 7.3: Beispielkonfiguration von /etc/mail/aliases

Im professionellen Einsatz besitzt dieses Setup zwei ausschlaggebende Vorzüge. Zum einen können Lizenzen für proprietäre Software eingespart werden. Es müssen nur jene Lizenzen für Software erworben werden, die auch wirklich verwendet werden müssen, alle anderen sind beispielsweise einfache POP3-Clients. Ein anderer Vorteil ist, daß der komplette SMTP-Traffic nicht direkt auf den Server für die interne Kommunikation trifft. Wäre hier ein Hackangriff auf dem durch den MX-Record bekannten Rechner erfolgreich, würde beispielsweise der MS-Exchange-Server auf einem ganz anderen Rechner unbeeinflußt weiterlaufen, so daß immerhin die In-House-Kommunikation ohne Schaden bliebe.

Listing 7.3 zeigt eine Beispieldatei »aliases.db«.

genericstable

Eine weitere Datei, die meiner Ansicht nach sehr interessant ist, ist »genericstable«. Sie ist für das Maskieren der Sender-Adresse des Mail-Envelopes bei sendmail verantwortlich. Befindet sich bei der installierten Sendmail-Version noch kein »genericstable(.db)« in »/etc/mail«, muß dieses Feature erst aktiviert werden. Standardmäßig ist dieses Feature jedoch eingeschaltet.

Listing 7.4 zeigt eine Beispieldatei.

```
# Beispielkonfiguration von /etc/mail/genericstable

# Senderadresse                 Adresse mit der der Sender
#                               maskiert werden soll.
thormy@localhost.localdomain    thorsten.s@web.de
thormy@localhost                thorsten.s@web.de
thormy                          thorsten.s@web.de
```

Listing 7.4: Eine Beispielkonfiguration von /etc/mail/genericstable

Daemon oder nicht?

Sendmail ist monolithisch in dem Sinne, daß die ganze Arbeit von einem Programm erledigt wird. Sendmail hat zwei Arbeitsweisen: es kann wie ein Programm aufgerufen werden, wenn es benötigt wird, oder es kann permanent im Hintergrund als Daemon laufen. Der Daemon-Modus wird nur dann benötigt, wenn die Aufgabe von Sendmail darin besteht, Nachrichten von externen Servern zu empfangen. Falls man dagegen Sendmail nur zum Versenden von Mails einsetzen möchte, sollte man nicht den Daemon-Modus benutzen. Allerdings könnten Sie dann auch mit dem Weiterlesen aufhören, denn man benötigt wirklich keine besonderen Einstellungen, um dies zu tun. Die Arbeitsweise von Sendmail hängt davon ab, wie es gestartet wurde. Wenn es als Daemon (mit dem Flag *-bd*) gestartet wurde, schaut es nach einkommenden SMTP-Verbindungen auf TCP-Port 25 und versucht immer wieder ausgehende Mails zu

versenden, die sich im Queue-Verzeichnis »/var/spool/mqueue« befinden. Ruft man dagegen Sendmail on the fly auf, versucht es, die auszuliefernden Nachrichten zu senden. Damit nun Sendmail automatisch als Daemon gestartet wird, muß man in der Datei »/etc/rc.conf« folgende Änderung machen:

```
sendmail=YES
```

Diese Einstellung ist unbedingt notwendig für die spätere Verwendung als SMTP-Gateway. Macht man dies nicht, werden zwar die lokalen Nachrichten verschickt, aber ansonsten muß man Sendmail immer per Hand aufrufen, damit ein Datentransfer stattfindet.

Das Ziel

Bevor wir weitermachen können, sollten wir festlegen, was wir eigentlich erreichen möchten. Ein SMTP-Gateway ist am lehrreichsten, weil es eine typische Rolle für Sendmail darstellt und weil gerade hier der Sicherheitsaspekt eine sehr große Rolle spielt. In den meisten Organisationen stehen öffentlich zugängliche Mailserver einer Unzahl von Bedrohungen gegenüber, mehr als dies interne Mail-Server tun, daher erfordert ein SMTP-Gateway typischerweise penibel genaue Sorgfalt bezüglich der Zugriffsstufen, Dateizugriffsrechte und ganz allgemein der Menge an Funktionen, da man gerade nur so viel Funktionen ermöglichen sollte, wie zum Routen von Mails absolut notwendig ist. Auf solch einem Host sollte Sendmail als unprivilegierter Benutzer arbeiten. Es sollte wenigstens *chroot*-et laufen, wenn Dateien zu schreiben sind, und es sollte eigentlich auch so konfiguriert sein, daß E-Mails nur für die eigene Organisation weitergeleitet werden, nicht jedoch für Spammer. Der Begriff »Spam« stammt aus einer Folge von Monty Python's Flying Circus, in dem es an Gerichten in einem Lokal nur Spam (also: Sülze) gab, jedoch stets in anderen Zusammenstellungen, aber halt immer nur Spam. Inzwischen hat es sich eingebürgert, die nervenden und unverlangten Werbesendungen, die stets in anderer Aufmachung kommen, als Spam zu bezeichnen.

Sendmail konfigurieren

Jetzt kommen wir dazu, Sendmail zu konfigurieren, so daß es als SMTP-Gateway für unsere Domain fungiert. Die vereinfachte Vorgehensweise zur Erstellung der Sendmail-Konfiguration (»sendmail.cf« und dazugehörige Dateien) besteht aus folgenden Schritten:

1. Aktivierung der benötigten Features in der »sendmail.mc«.
2. Einrichtung des Domain-Name Masquerading in »sendmail.mc«, falls nötig.
3. Starten von *m4*, um »sendmail.cf« aus »sendmail.mc« zu erzeugen.
4. Konfigurierung der Zustellungsregeln durch Editieren von »mailertable«.
5. Konfigurierung der Weiterleitungsregeln durch Editieren von »access«.
6. Konfigurierung der lokalen Benutzer-Aliase in »aliases«.

7. Konvertierung von »mailertable«, »access« und »aliases« ins Datenbank-Format.
8. Definieren aller lokalen Host-Namen in »local-host-names«.
9. Starten von Sendmail.

Es würde den Rahmen dieses Buchs sprengen, die Syntax der Konfigurationsdatei »sendmail.mc« im Detail zu beschreiben. Deswegen beschränke ich mich auf einige sinnvolle Aspekte bezüglich der Sicherheit und der Modularisierung unserer Konfiguration. Zusätzlich zur »sendmail.cf« selbst ist es möglich, Sendmail so konfigurieren, daß es die Konfigurationsdaten aus mehreren unterschiedlichen Dateien liest, was aus zwei Gründen zweckmäßig ist. Zuerst ist es sehr unangenehm, »sendmail.cf« direkt zu editieren und auch die Neu-Erstellung der »sendmail.cf« aus der »sendmail.mc« ist nicht immer wünschenswert. Als Zweites, falls Ihr SMTP-Gateway mehrere Verwalter mit variierenden Rechten besitzt, wäre es erstrebenswert, »sendmail.mc« und »sendmail.cf« vor deren Zugriffe zu sperren, aber trotzdem einigen Administratoren die Möglichkeit zu geben, die Benutzer-Aliase, Mail-Zustellungsregeln und so weiter editieren oder modifizieren zu können.

Die wichtigsten externen Konfigurationsdateien, die es zu aktivieren gilt, sind »mailertable«, »virtusertable« und »access«. Sie wurden bereits neben anderen wichtigen Dateien vorgestellt. Die »sendmail.mc«-Anweisungen, um die Dateien zu aktivieren, sind in Listing 7.5 gezeigt.

```
FEATURE('mailertable', 'hash -o /etc/mail/mailertable.db')dnl
FEATURE('virtusertable', 'hash -o /etc/mail/virtusertable.db')dnl
FEATURE('access_db', 'hash -o /etc/mail/access.db')dnl
```

Listing 7.5: Dateiaktivierung in der sendmail.mc

Jeder dieser Zeilen in Listing 7.5 teilt Sendmail mit, sich auf eine spezifizierte Datei zu beziehen, eine Hash-Datenbank zu benutzen und selbstverständlich auch den Pfad, wo sich die jeweilige Datei befindet. Wir werden in Kürze sehen, wie man mit diesen Dateien arbeitet, aber zuerst müssen wir noch ein paar Sachen in der »sendmail.mc« näher betrachten.

Falls die E-Mail-Adresse der Benutzer generisch mit deren Domain ist und zwar eher als zu bestimmten Hosts, in die sie sich einloggen, also beispielsweise eher *thorsten@mydomain.com* als *thorsten@trunks.mydomain.com*, wollen Sie sicherlich auch, daß sich dieser Umstand im Von-Feld (from-field) der ausgehenden E-Mail widerspiegelt. Der Empfang von E-Mails mit solchen generischen Namen erfordert die Verwendung von Benutzer-Aliasen.

Die folgenden Zeilen der »sendmail.mc« zeigen, wie man das Von-Feld des *mydomain.com*-Benutzers wie gewünscht ändert. Alle Zeilen in Listing 7.6 müssen hinzugefügt (oder eventuell nur unkommentiert werden).

```
MASQUERADE_AS(`mydomain.com')dnl
MASQUERADE_DOMAIN(`mydomain.com')dnl
EXPOSED_USER(`root')dnl
FEATURE(`masquerade_entire_domain')dnl
FEATURE(`masquerade_envelope')dnl
```

Listing 7.6: Das Ändern des Von-Felds

Zu Listing 7.6 sei mir noch eine Bemerkung gestattet. Es handelt sich bei den Markierungszeichen »`« und »'« um keine Schreib- oder Druckfehler. Eine andere Sache, an die sich das Erinnern lohnt, ist die Art und Weise, wie Kommentare in der »sendmail.mc« geschrieben werden, da man hierfür das Kommentierungswort »dnl« und nicht »#« oder wie in C üblich »/* */« verwendet.

Die Anweisung *MASQUERADE_AS* bestimmt den vollqualifizierten Domainnamen (FQDN = Fully Qualified Domain Name), der in der Von-Adresse erscheinen soll. Die Anweisung *MASQUERADE_DOMAIN* spezifiziert dagegen die Hosts für die *MASQUERADE_AS* anwendbar ist. Beachten Sie bitte den vorangehenden ».« (Punkt) vor *mydomain.com*, dieser zeigt an, daß alle Hostnamen mit diesem Domain-Namen zu maskieren sind.

EXPOSED_USER spezifiziert einen Benutzernamen, für den die Von-Adresse nicht maskiert werden soll. Der Benutzer mit Namen *root*, also der Systemadministrator, ist ein häufig benutzter Kandidat hierfür, da E-Mails vom *root* des öfteren Warnungen sowie Alarmierungen enthalten. Erhält man also eine solche E-Mail, möchte man im allgemeinen schon wissen, von welchem Host sie gesendet wurde.

Das Merkmal *masquerade_entire_domain* veranlaßt, *MASQUERADE_DOMAIN* als ganze Domain anzusehen und nicht nur als Hostnamen. Das Merkmal *masquerade_envelope* wendet die Maskierung nicht nur auf den SMTP-Header an, sondern auch auf den E-Mail-Envelope.

```
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(`smrsh',`/usr/sbin/smrsh')dnl
define(`confSAFE_FILE_ENV', `/var/mailjail')dnl
```

Listing 7.7: Zusätzliche Features

Vier weitere Anweisung werden in Listing 7.7 vorgestellt, diese erläutere ich hier kurz, weil sie recht wichtig sind. Das Feature *always_add_domain* erzwingt, den Domainnamen des *localhosts* an jede E-Mail, die ursprünglich von einem Host kommt, der sich selbst ohne Domainnamen identifiziert, anzuhängen.

Wenn beispielsweise das SMTP-Gateway eine E-Mail vom Benutzer *aragon* von einem Host, der sich als *ringpire* identifiziert, empfängt, schreibt Sendmail das Von-Feld als *aragon@ringpire.mydomain.com* um. Aber natürlich werden die Maskierungsanweisungen weiterhin angewendet.

Die Eigenschaft *use_cw_file* bringt Sendmail dazu, sich auf die Datei »/etc/mail/local-host-names« zu beziehen, in der die Hostnamen (einer pro Zeile) stehen, die als lokal angesehen werden sollen. Angenommen, unser Beispiel-SMTP-Gateway empfängt E-Mails nicht nur für *mydomain.com*, sondern auch für *zuhause.net*. Wenn nun der Hostname des Gateways *mail* ist, wird die »local-host-names«-Datei wie in Listing 7.8 aussehen:

```
localhost.localdomain
mail.mydomain.com
mail.zuhause.net
```

Listing 7.8: Zuweisung des Gateways

Die dritte aktivierte Eigenschaft ist *smrsh*, die »sendmail restricted shell«. Sie ist eine sehr wichtige Sicherheitskontrolle, da sie die Befehle beschränkt, die von der ».forward«-Datei eines Benutzers ausgeführt werden dürfen.

Die vierte Zeile von Listing 7.7 fordert Sendmail auf, die »sendmail.cf«-Variable *SafeFileEnvironment* auf – Sie raten es sicherlich schon – ein Unterverzeichnis von »/« zu setzen, das Sendmail chrooten wird. Tatsächlich passiert dies nur, wenn Sendmail Dateien schreibt. Wenn man richtig darüber nachdenkt, macht dieses fünfzigprozentige *chroot* Sinn: wir machen uns ja die meisten Sorgen über das Schreiben von Dateien. Das Anlegen einer *chroot*-Umgebung auf diese Art und Weise ist wesentlich einfacher als die typischen chroot-jails, die Kopien von allem enthalten müssen, was unser chrootet Programm benötigt.

Sendmail legt die Dateien »/var/mailjail/var/spool/mqueue/aragon« und ».../root« an. Zuvor hatte ich jedoch das Mail-Gefängnis mit nur vier Befehlen erzeugt (siehe Listing 7.9).

```
mkdir -p /var/mailjail/var/spool/mail /var/mailjail/var/spool/mqueue
cd /var/mailjail
chown -R mail:mail *
chmod -R 700 *
```

Listing 7.9: Das Anlegen des Mail-Gefängnisses

Falls Sie sich um Spammer sorgen, habe ich eine gute Nachricht für Sie. Standardmäßig erlaubt Sendmail kein SMTP-Relay, was ja eine übliche Technik ist. Das können Sie

auch in der »sendmail.mc« deaktivieren, aber ich werde Ihnen nicht verraten, wie das geht. Lassen Sie diese Einstellung einfach in Ruhe. Ferner kann man Sendmail dazu bewegen, Nachrichten von bekannten Spammern abzuweisen (*REJECT*) und zwar durch die Realtime Blackhole List (kurz: RBL), was ein großartiges Feature ist. Statt der Einführung eines neuen Protokolls wird auf die bewährte DNS-Technologie gesetzt. Eine öffentlich gepflegte Liste mit dem Namen *RBL* wird bei jeder Nutzung des Sendmail-Services zu Rate gezogen. Ist der Benutzer in dieser Liste als Spammer verzeichnet, wird der weitere Handlungsverlauf von Sendmail aktiv beeinflußt. Um den RBL-Dienst zu aktivieren, fügt oder kommentiert man einfach folgende Zeile ein oder aus:

```
FEATURE(`dnsbl')
```

Nach der Eingabe muß man wieder aus der »sendmail.mc« die »sendmail.cf« erstellen lassen. Am Ende muß nun noch der *sendmail*-Dienst neu gestartet werden.

Um neue Spammer in die Liste eintragen zu lassen, muß die Person beziehungsweise deren E-Mail-Adresse den Betreuern der RBL gemeldet werden, die dann die notwendigen Einstellungen vornehmen, die sich sofort auch auf den Betrieb des eigenen SMTP-Servers auswirken. Mehr Informationen zu solchen Reports finden sich in der FAQ unter *http://mail-abuse.org/rbl/notifyfaq.html*.

Die Benutzung der RBL kann möglicherweise sowohl die E-Mail von legitimen Benutzern als auch von Spammern blockieren, deshalb sollte man mit Vorsicht an die Sache gehen.

Bitte achten Sie darauf, daß das folgende Feature in der »sendmail.mc« deaktiviert ist:

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
```

Dies nimmt man am besten vor, indem man die entsprechende Zeile auskommentiert. Das Auskommentieren funktioniert wie bereits erwähnt, indem man vor die Zeile das Wort »dnl« schreibt. Dies ist wichtig, da diese Zeile sonst nur Verbindungen vom *loopback*-Interface zuläßt, jedoch nicht von externen Hosts. Es ist unnötig, darauf hinzuweisen, daß das für ein SMTP-Gateway nicht erstrebenswert ist.

Das waren die wichtigsten »sendmail.mc«-Einstellungen für unsere Zwecke. Es gibt andere relevante Einstellungen in Bezug auf Sicherheit, speziell für das Nicht-Gateway-Szenario (lokale Zustellung und so weiter). Für weitere Informationen schauen Sie bitte in die »README.cf«.

Um die Makro-Konfigurationsdatei in die »sendmail.cf« zu übersetzen, benutzt man folgenden Befehl:

```
m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
```

Wie man sich vielleicht schon denken kann, ist der Name »sendmail.mc« nicht unbedingt wichtig. Diese Makro-Datei kann man nennen wie man möchte. Sendmail erwartet als Konfigurationsdatei stets eine Datei mit Namen »sendmail.cf«, darauf ist unter allen Umständen zu achten.

Anmerkung:

Im Verzeichnis »/usr/share/sendmail/cf« befinden sich die Dateien »freebsd-proto.mc«, mit der die mitgelieferte »sendmail.cf« generiert wird. Mit dem *make*-Befehl kann sie wieder neu erstellt werden, falls das nötig ist.

Konfigurierung der Zustellungsregeln (delivery rules)

Das war nun der schwere Teil. Nun müssen wir Sendmail nur noch mitteilen, was es mit den eingehenden Nachrichten tun soll, welche lokalen Hostnamen legitim sind und welche Benutzer, Netzwerke und Domains das SMTP-Gateway zum Weiterleiten von Nachrichten, die nicht lokal sind, benutzen darf.

Die Datei »mailertable« definiert solche Zustellungsregeln. Sie besitzt eine einfache Syntax. Jede Zeile besteht aus zwei Teilen: einem Identifizierer für das Ziel und einer Aktion. Der Ziel-Identifizierer stimmt mit der Ziel-Adresse oder Teilen davon überein. Die Aktion sagt Sendmail, was sie mit den E-Mails machen soll, deren Ziel mit dem Identifizierer übereinstimmt.

Falls der Identifizierer mit ».« beginnt, stimmen alle E-Mail-Ursprungsadressen, was auch immer dem Punkt folgt, überein. Falls nicht, muß alles, was nach dem »@« kommt, identisch mit dem Identifizierer sein. Die E-Mail-Adresse *aragon@stupido.mydomain.com* wird also nicht mit dem Identifizierer *mydomain.com* übereinstimmen, jedoch mit dem Identifizierer *.mydomain.com*.

Die Aktion hat die Form *agent:action*, wobei *agent* entweder ein Mailer (definiert in »sendmail.mc« im Ausdruck *MAILER()*) oder die eingebauten Agenten *local* oder *error* ist. Der *local*-Agent bedeutet natürlich, daß die E-Mail einem lokalen Benutzer zugestellt werden soll, der nach dem Doppelpunkt spezifiziert wurde. Falls nichts nach dem Doppelpunkt folgt, wird der in der E-Mail spezifizierte Benutzer verwendet.

In Listing 7.10 finden Sie zwei unterschiedliche Aktionen.

```
mydomain.com          smtp:mailmonster.mydomain.com
mail.mydomain.com     local:postmaster
```

Listing 7.10: Mailagenten

Zusätzlich zu den Zustellungsregeln (Delivery Rules) muß Sendmail ferner wissen, welche E-Mail-Ziel-Adressen als lokal angesehen werden sollen. Sie werden in der »/etc/mail/local-host-names« genauer bestimmt, wobei pro Zeile jeweils immer nur ein Hostname stehen darf.

Zuletzt müssen wir noch die Weiterleitungsregeln erstellen, indem wir die »/etc/mail/access« editieren. Die Syntax ist einfach. Jede Zeile besteht aus einem Quell-Namen oder einer Adresse, gepaart mit einer Aktion. Die Aktion kann *RELAY*, *REJECT*, *DISCARD*, *OK* oder *ERROR* sein. An dieser Stelle erläutere ich die eben genannten Aktionen nochmals kurz:

RELAY	Zulassen der Weiterleitungsfunktion (relay).
OK	Eine E-Mail wird auch dann zugelassen, falls eine andere Regel dagegen spricht.
REJECT	Abblocken von unerwünschten E-Mails mit serverseitiger Standard-Meldung.
DISCARD	Die E-Mails werden verworfen, dies bedeutet, daß keine serverseitigen Status-Meldungen an den Benutzer geschickt werden. Dies kann bei vermeintlichen Spam-Angriffen hilfreich sein, um den Angreifer im Ungewissen zu lassen. Außerdem ist es auch möglich, individuelle Fehlermeldungen nach den Regeln der RFC 821-Fehlerantwortcodes auszugeben.

In der Praxis ist die nützlichste Aktion wohl *RELAY*, da standardmäßig das Weiterleiten abgeblockt (*REJECT*) wird. *REJECT* und *DISCARD* sind nur dann sinnvoll, wenn es darum geht, Ausnahmen zu expliziten *RELAY*-Regeln zu definieren.

Den Inhalt einer einfachen »access«-Datei finden Sie in Listing 7.11.

```
localhost.localdomain          RELAY
localhost                      RELAY
192.168                        RELAY
127.0.0.1                      RELAY
```

Listing 7.11: Eine einfache access-Datei

Haben Sie das Fehlen von echten Hostnamen im Beispiel (Listing 7.11) bemerkt?

In diesem Beispiel trifft das SMTP-Gateway nur Weiterleitungen nach außen. Eingehende Nachrichten müssen an eine lokale E-Mail-Adresse adressiert sein. Ausgehende E-Mails müssen von Hosts kommen, deren IP mit 192.168 (offensichtlich einem nicht internet-routbaren Netzwerk) beginnt.

Tips zur Sicherheit

Die Sicherheitsstandards von Sendmail wurden in den letzten Jahren stark erhöht, trotzdem ist dieser Dienst immer noch eines der begehrtesten Angriffsziele für Hakker. Hierfür gibt es zwei Hauptgründe:

- Sendmail ist ein öffentlich verfügbarer Service. Wenn er gestartet ist, kann sich eigentlich jeder mit ihm verbinden.
- Sendmail läuft im allgemeinen als *root*, daher werden erfolgreiche Attacken zumeist mit Administrator-Rechten belohnt.

Bitte achten Sie regelmäßig auf den entsprechenden Seiten wie zum Beispiel *www.sendmail.org* auf neue Security-Patches!

Gefährliche Befehle

Es gibt zwei SMTP-Standardbefehle, die von Spammern (oder auch Hackern) gern eingesetzt werden, um legitime Benutzer auf einem System zu identifizieren. Es handelt sich dabei um *EXPN* und *VRFY*. Der Expand-Befehl *EXPN* wird zur automatischen Vervollständigung von Mailing-Listen angewandt. Der Verify-Befehl *VRFY* hingegen dient zur automatischen Vervollständigung von Benutzernamen. Falls möglich, sollten diese beiden Funktionen in der »sendmail.cf« durch den Eintrag *PrivacyOption* deaktiviert werden, der geänderte Teil der »/etc/sendmail.cf« sieht dann wie folgt aus:

```
O PrivacyOptions=authwarnings,noexpn,novrfy
```

Um die Änderungen zu aktivieren, ist der Sendmail-Service durch die Eingabe von *»/etc/rc.d/sendmail restart* neuzustarten. Eine Überprüfung dieser Einstellungen und der neu gewonnen Sicherheit kann beispielsweise mit einer Telnet-Verbindung auf Port 25 vorgenommen werden. Damit kann man testen, ob die nun verbotenen Befehle immer noch nutzbar sind.

8 Wählverbindungen

von Karl Uwe Lockhoff

Moderne Rechnernetzwerke sind heutzutage nicht nur lokal, sondern weltweit miteinander im LAN und WAN verbunden. Wer mit seinem NetBSD in das Internet will und keinen Anschluß an eine Standleitung hat, sollte dieses Kapitel lesen.

Mit LAN (Local Area Network) sind Verbindungen gemeint, die nur eine verhältnismäßig kurze Reichweite besitzen. Zu den Verbindungstechniken im LAN gehört beispielsweise das Ethernet. Bei allen Techniken im LAN stehen Performance und Preis im Vordergrund, es werden meist mehrere Rechner physikalisch an einer Leitung angeschlossen.

Im WAN (Wide Area Network) wird die Reichweite erhöht. Da die Übertragung eines einzelnen Pakets im WAN meist länger dauert als im LAN, ist die Vorgehensweise hier anders. Es kommunizieren immer zwei Rechner direkt miteinander, die Leitung steht also beiden Rechnern immer zur Verfügung. Aufgrund der im Vergleich zum LAN hohen Anfälligkeit für Fehler (beispielsweise bei einer schlechten Telefonleitung) spielt die Kontrolle der Qualität, die Fehlererkennung und die Fehlerkorrektur eine größere Rolle. Deshalb steht bei den WAN-Protokollen auch die Kontrolle der Verbindungsqualität, die Fehlererkennung und die Fehlerkorrektur im Vordergrund.

Bei den WAN-Techniken unterscheidet man weiterhin zwischen permanenten Verbindungen wie beim LAN und Verbindungen bei Bedarf. Von letzteren ist im folgenden die Rede, denn es werden hier Telephonieverfahren beschrieben.

Solange nicht anders erwähnt, werden die Kernelergänzungen für den PC-Port (i386) angegeben.

WAN-Protokolle für Dial-up-Verbindungen

Heute finden hauptsächlich drei Protokolle Verwendung:

- ppp: Analoge Telefonleitung oder direkte Verbindung zweier Rechner
- ISDN: Digitale Telefonleitung
- DSL: Breitbandverfahren für konventionelle Leitungen

Um einen Rechner erfolgreich mit der Internet zu verbinden, müssen einige Voraussetzungen erfüllt sein:

1. Die Telefonnummer des Einwahlpunkts muß bekannt sein.
2. Man muß Kenntnis von der Autorisierungsmethode haben.
3. Die Benutzerkennung und das Paßwort zur Einwahl müssen in Erfahrung gebracht werden.
4. Man muß die IP-Nummer des Name-Servers wissen.

Mit diesen Wissen kann die Basiskonfiguration für den Zugang ins Internet beginnen.

8.1 Mit ppp ins Internet

8.1.1 Kernel-Anforderungen

Ein zur Zeit übliches Verfahren für die Einwahl ins Internet ist der Zugang über analoge Modems mit ppp. Für dieses Verfahren werden unter NetBSD ein paar Anforderungen an den Kernel gestellt. Es ist also zu überprüfen, ob folgende Kerneloptionen (für alle Ports) eincompiliert sind:

```
options       PPP_BSDCOMP       # Unterstützung für Kompression
options       PPP_DEFLATE       # Eine weitere Kompression
options       PPP_FILTER        # Aktivierung des Filters
                                # bpf-Unterstützung im Kernel
                                # Hierfür ist erforderlich:
pseudo-device  bpfilter    8    # Berkeley Packet Filter
pseudo-device  ppp         2    # Schnittstelle für ppp
```

Sollten diese Optionen nicht im Kernel vorhanden sein, muß er um diese Optionen erweitert, konfiguriert, erzeugt und installiert werden.

Anmerkung

Das Kommando *ifconfig -C* listet alle bekannten Interfaces in einem Kernel auf.

8.1.2 Wichtige Konfigurationsdateien

Einzelne Rechner werden fast ausschließlich über Namen angesprochen. Die Netzwerksoftware selbst verwendet aber immer die sogenannte IP-Nummer. Dies bedeutet, daß Namen in IP-Nummern umgewandelt werden müssen. Dies bezeichnet man auch als Auflösen des Namens, Internet Domain Name Service oder kurz DNS. Da man bei Dial-up-Verbindungen keine permanente Verbindung zum Internet und damit auch keine Verbindung zu einem hierfür zuständigen Server, dem Name-Server hat, sind ein paar Maßnahmen erforderlich:

/etc/hosts

Diese Datei stellt eine lokale Datenbank für bekannte Rechner da. Jede Zeile in dieser Datei enthält die IP-Nummer eines Rechners, den Namen und möglicherweise noch dessen Aliases. Mit Hilfe dieser Datei könnten Netzwerke ohne Name-Server konfiguriert werden. Das ist aber nur für wirklich kleine Netze brauchbar. Was man aber immer machen sollte, ist den Rechner selber in dieser Datei einzutragen, damit er sich im Notfall immer selbst finden kann:

```
# Beispiel einer /etc/hosts
127.0.0.1       localhost
198.162.1.1     gandalf
```

Die Zeile mit dem Eintrag *localhost* ist immer erforderlich. Über diesen Namen und diese IP-Nummer verwaltet jeder Rechner sein internes Netzwerk.

Der zweite und eventuelle weitere Einträge sind nur für den Fall erforderlich, wenn sich der Rechner in einem LAN befindet.

/etc/resolv.conf

Hier werden die Name-Server, mit denen die Namen aufgelöst werden, konfiguriert:

```
# Beispiel einer /etc/resolv.conf
server 134.134.145.41
```

Es wird der Name-Server des Providers eingetragen.

/etc/nsswitch.conf

Da verschiedene Quellen zur Auflösung von Internetnamen verwendet werden, muß es auch eine Möglichkeit geben, diese auswählen zu können. Dafür dient die Datei »/etc/nsswitch.conf«. Für Dial-up-Verbindungen hat sich folgende Zeile in dieser Datei bewährt und sollte entsprechend angepaßt werden:

```
host: files dns
```

Da Dial-up-Verbindungen nicht immer bestehen, sollte immer versucht werden, zuerst in der »/etc/hosts« (Eintrag *files*) und dann erst über DNS den Namen aufzulösen. Wäre dies anders herum, müßte erst eine Internet-Verbindung aufgebaut werden, bevor ein Name aufgelöst werden kann. Da NetBSD beziehungsweise einige Anwendungen auch mal nach dem Namen des lokalen Rechners fragen, kann dies auf diese Weise einfach verhindert werden.

Anmerkung

Diese grundlegende Konfiguration gilt für alle Dial-up-Verbindungen.

Anlegen der Verzeichnisse für ppp-Konfigurationsdateien

Die Verzeichnisse »/etc/ppp« und »/etc/ppp/peers« enthalten unter NetBSD die Konfigurationsdateien für ppp. Diese Verzeichnisse sind nach einer frischen NetBSD-Installation nicht vorhanden und müssen angelegt werden:

```
# mkdir /etc/ppp
# mkdir /etc/ppp/peers
# chmod 700 /etc/ppp
# chmod 700 /etc/ppp/peers
```

Die Änderung der Zugriffsrechte erfolgt aus Sicherheitsgründen. Die Konfigurationsdateien werden Informationen wie Benutzerkennungen und Paßworte im Klartext enthalten, die ja nicht jeder lesen können soll.

Beim späteren Einwählen wird das Programm *pppd*, der ppp-Daemon, gestartet. Dieses Programm braucht als Parameter das sogenannte Verbindungsskript (engl. connect script). Dieses Skript wird unter »/etc/ppp/peers« abgelegt und trägt üblicherweise den Namen des Providers:

```
# Beispiel /etc/ppp/peers/weitewelt
connect `/usr/sbin/chat -v -f /etc/ppp/peers/weitewelt.chat`
noauth
user weltzugang
remotename weitewelt.de
```

In diesem Beispiel wird ein weiteres Skript */etc/ppp/peers/weitewelt.chat* verwendet. Dieses Skript (englisch: chat script) übernimmt in der Anfangsphase des Verbindungsaufbaus die Unterhaltung mit dem Modem und dem Provider.

```
# /etc/ppp/peers/weitewelt.chat
ABORT     BUSY
ABORT     'NO CARRIER'
ABORT     'NO DIALTONE'
REPORT    CONNECT
SAY       'Baue Verbindung zur weiten Welt'
~         ATDT 0123456789
CONNECT   ~
```

In diesem Skript werden Regeln für verschiedene Zustände definiert. So soll der Versuch abgebrochen werden, falls die Leitung besetzt ist (*BUSY*), man keine Leitung bekommt (*NO CARRIER*) oder kein Amtsignal erhält (*NO DIALTONE*). In allen anderen Fällen wird *CONNECT* berichtet werden. Die Zeile *SAY* soll über den Versuch des Verbindungsaufbaus berichten. ATDT 0123456789 veranlaßt das Modem nun zur Wahl. Hier ist die Telefonnummer des Providers anzupassen. Die letzte Zeile beendet die Unterhaltung erfolgreich.

Anmerkung

1. Fügt man in das Verbindungsskript die folgenden zwei Zeilen hinzu, erhält man Log-Ausgaben zur Problemidentifizierung:

   ```
   debug
   kdebug 4
   ```

   Für weitere Infomationen hierzu siehe pppd(8) und syslog.conf(5).
2. Bei Problemen mit der Unterhaltung mit dem Provider kann man mit dem Programm *cu* die Verbindung manuell aufbauen und sich den Dialog angucken. Siehe *cu*(1).
3. Bequemer als *cu*(1) sind einige Programme aus den Packages, zum Beispiel *minicom* oder *kermit*.

Nach der Einwahl versuchen beide Systeme, sich zu identifizieren. Genaugenommen müssen beide Stellen sich gegenseitig anerkennen. In der Praxis beim Umgang mit Internetprovidern wird aber der Provider vom Einwählenden einfach akzeptiert und nur der Einwählende hat sich beim Provider auszuweisen. Dafür wird eine der folgenden Methoden verwendet:

1. login
2. PAP
3. CHAP

Die Informationen für diese Form der Authentifizierung werden in den Dateien »/etc/ppp/pap-secret« beziehungsweise »/etc/ppp/chap-secrets« gespeichert:

```
# Beispiel /etc/ppp/pap-secret bzw. /etc/ppp/chap-secret
# Benutzer * Passwort
weltzugang * Dkjl43
```

Aus Sicherheitsgründen sollte *root* Eigentümer dieser Datei sein und die Zugriffsrechte sollten sich auf ihn beschränken (600).

login

Bei dieser Methode ist das chat-Skript einfach zu erweitern:

```
# /etc/ppp/peers/weitewelt.chat
ABORT     BUSY
ABORT     "NO CARRIER"
ABORT     "NO DIALTONE"
REPORT    CONNECT
SAY       "Baue Verbindung zu weitewelt"
```

```
~         ATDT 0123456789
CONNECT   ~
TIMEOUT   50
ogin:     weltzugang
assword:  Dkj143
```

Manche Provider haben hier ein wenig gedreht, beispielsweise *Paßwort* anstelle von *password*. Dies kann jedoch leicht angepaßt werden.

pppd-Optionen

Bisher war vom Modem noch nicht viel die Rede. Dies soll jetzt anders werden. Damit der *pppd* das Modem findet, wird eine Datei »/etc/ppp/options«, die Optionendatei, notwendig:

```
/dev/ttyO1
lock
crtscts
57600
modem
defaultroute
noipdefault
```

Die Bedeutung der einzelnen Zeilen der Reihe nach:

1. Das Gerät (englisch: device), mit dem das Modem verbunden ist.
2. Erzeugen einer Lock-Datei, um andere Software an der Nutzung des Modems zu hindern.
3. Die Flußkontrolle festlegen.
4. Die Geschwindigkeit auf der seriellen Leitung.
5. Am Gerät ist ein Modem angeschlossen.
6. Die Verbindung soll die Standardroute werden.
7. Es wird keine IP-Nummer vorgegeben. Der Rechner wird die Nummer vom Provider beziehen.

Eine genauere Beschreibung dieser und weiterer Optionen wird unter man(8) gegeben.

Jetzt kann ein erster Test erfolgen:

```
# pppd call weitewelt
```

Mit dem Befehl *tail -f /var/log/messages* kann man den Verbindungsaufbau verfolgen. Die Verbindung wird mit *kill -HUP* auf den pppd wieder getrennt.

Einfache Verbindungen auf- und abbauen

Um den Verbindungsaufbau zu vereinfachen, kann man sich einen Satz von Skripten schreiben. Das Skript zur Einwahl sieht so aus:

```
#!/bin/sh
# ppp-up: Skript zum verbinden mit dem Internet
#
# Zuerst wird der Zustand der serielle Schnittstelle
# überprüft. Um anzuzeigen, daß die serielle Schnittstelle
# belegt ist, können Programme wie der pppd eine
# Datei LCK.gerätename im Verzeichnis /var/spool/lock
# anlegen. Typischerweise enthält diese Datei die PID
# des laufenden Programms, das die Schnittstelle belegt.

if [ -f /var/spool/lock/LCK..ttyO1 ]; then
   echo Der pppd läuft bereits.
else
   ppp call weitewelt
# Hier können weitere Aktionen gestartet werden
   fetchmail
fi
```

Das folgende Skript trennt die Verbindung wieder:

```
#!/bin/sh
# ppp-down: Skript zum Trennen der Verbindung

if [ -f /var/spool/lock/LCK..ttyO1 ]; then
   echo -n Der pppd wird gestoppt.
   kill -HUP 'cat /var/spool/lock/LCK..ttyO1`
   echo Fertig.
else
   echo Der pppd läuft nicht.
fi
```

Jetzt muß man nur noch die Skripte ausführbar machen (*chmod u+x ppp-up ppp-down*) und fertig. Durch *ppp-up* und *ppp-down* kann die Einwahl ins Internet jetzt einfach gehandhabt werden.

Automatische Einwahl und Trennung

Anstatt die Einwahl von Hand über die Skripte durchzuführen, können unter NetBSD auch die Verbindung und die Trennung automatisch erfolgen. Hierzu wird das bisherige Setup nur ein wenig erweitert:

1. Das Schlüsselwort *demand* wird der Optionsdatei »/etc/ppp/options« hinzugefügt.
2. Der pppd reagiert auf Pakete, die in eine bestimmte Richtung versendet werden, daher wird die IP-Nummer des Einwahlrechners noch als Ziel in die Optionsdatei aufgenommen. Diese IP-Nummer ist beim Provider zu erfragen oder zu ermitteln.
3. Damit ein automatischer Abbruch der Verbindung erfolgt, ist eine Zeitspanne in Sekunden festzulegen und nach dem Schlüsselwort *idle* ebenfalls in die Optionsdatei einzutragen.

Die Optionsdatei »/etc/ppp/options« sollte jetzt so aussehen:

```
/dev/ttyO1
lock
crtscts
57600
modem
defaultroute
noipdefault
demand
idle 5
:123.123.234.1
```

Damit der pppd jetzt noch beim Booten mitgestartet wird, ist die Konfiguration in der »/etc/defaults/rc.conf« anzupassen:

```
pppd=YES pppd_flags="call weitewelt"
```

Nach einem Reboot sollte das System sich jetzt bei Bedarf einwählen und die Verbindung auch wieder automatisch trennen.

8.2 Ein Netzwerk mit dem Internet verbinden

Häufig besitzt man heutzutage nicht nur einen Rechner, sondern ein komplettes Netzwerk. Manchmal ist es erforderlich, ein komplettes Netzwerk, zum Beispiel ein Firmennetz, mit dem Internet zu verbinden. Hier gibt es zwei wichtige Fälle:

1. Das Netzwerk ist dem Provider bekannt. Im diesem Fall reicht ein übliches Netzwerksetup aus.
2. Das Netz ist dem Provider nicht bekannt oder der Provider vergibt nur eine einzelne IP-Nummer. Diese Situation wird im folgenden beschrieben.

Der Rechner mit der Verbindung zum Internet über ppp wird als »Gateway« bezeichnet, alle anderen Rechner des Netzwerkes als Clients.

Zuerst ist das übliche Netzwerksetup wie gewohnt vorzunehmen. Jedem Rechner ist eine IP-Nummer für das Netzwerkinterface, normalerweise Ethernet, zuzuweisen. Die IP-Nummern sollten aus den für private Netze reservierten Bereichen 10.0.0.0/24 oder 192.168.0.0/16 stammen. Die IP-Nummer des Gateways ist in die Datei »/etc/mygate«

einzutragen. Der Einfachheit halber werden alle Rechnernamen des lokalen Netzwerks samt IP-Nummer in die »/etc/hosts« eingetragen und auf allen Rechnern installiert. Die Änderungen für »/etc/nsswitch.conf« sind wie oben beschrieben vorzunehmen. Die »/etc/resolv.conf« ist mit der gleichnamigen Datei des Gateways identisch. Nach diesen Maßnahmen sollten die Rechner miteinander kommunizieren können.

Auf dem Gateway ist alles ähnlich. Der einzige Unterschied ist, daß die Datei »/etc/mygate« hier nicht vorhanden ist und das LAN-Interface mit einer Netzmaske definiert ist.

Nun sind noch einige zusätzliche Kleinigkeiten auf dem Gateway zu erledigen. Der Kernel sollte mit folgender Zeile konfiguriert werden:

```
pseudo-device   ipfilter   # IP-Pakete werden hierdurch gefiltert und
                           # können bei Bedarf manipuliert werden.
```

Danach sollte die Kernelvariable *net.inet.ip.forwarding* überprüft werden, damit der IP-Verkehr zwischen der einzelnen Interfaces weitergeleitet wird:

```
# sysctl net.inet.ip.forwarding
net.inet.ip.forwarding = 0
```

Ist der Wert der Variablen wie oben Null, muß dieser Wert geändert werden:

```
# sysctl -w net.inet.ip.forwarding=1
```

Damit diese Angabe gleich bei Booten gesetzt ist, sollte der Eintrag in der Datei »/etc/sysctl.conf« erfolgen:

```
net.inet.ip.forwarding=1
```

Alternativ kann es direkt in den Kernel aufgenommen werden:

```
options         gateway
```

Der Rest dieses Abschnitts widmet sich dem Thema IPNAT (Internet Protocol Network Address Translation), also der Umsetzung der internen IP-Nummer. Diese intern verwendeten Nummern sind im Internet nicht bekannt und daher nicht erreichbar, weswegen sie umgesetzt werden müssen. Dafür wird eine Datei »/etc/ipnat.conf« angelegt:

```
map ppp0 192.168.1.0/24 -> 0/32 proxy port ftp ftp/tcp
map ppp0 192.168.1.0/24 -> 0/32 portmap tcp/udp 40000:60000
map ppp0 192.168.1.0/23 -> 0/32
```

Zur Erläuterung: 192.168.1.0/24 ist das innere Netzwerk, also die IP-Nummer, die verändert werden soll. 0/32 ist eine Umschreibung für einen einzelnen Rechner mit unbekannter IP-Nummer. Diese IP-Nummer wird vom Provider ja erst bei der Einwahl zugewiesen.

Nun zu den einzelnen Zeilen:

1. Hiermit wird nur der ftp-Service durch das Gateway ermöglicht. Durch die Natur des ftp-Protokolls muß der Betrieb über einen Proxy abgewickelt werden.
2. Die Basis-Protokolle tcp und udp sollen korrekt arbeiten. Das Verändern der Portnummern ist notwendig, da ja nur ein Rechner im Internet bekannt ist. Damit es nicht zu Konflikten mit Portnummern auf den Clients kommt, werden die Ports umgesetzt.
3. Alle restlichen Protokolle werden ebenfalls umgesetzt, damit auch ICMP für *traceroute*&Co. möglich ist.

Für weitergehende Beispiele können die Dateien im Verzeichnis »/usr/share/doc/ipf« studiert werden.

Wenn die Verbindung über ppp besteht, sollte die Umsetzung aktiviert und beim Auflegen deaktiviert werden. Um dies zu bewerkstelligen, sind zwei Skripte zu schreiben:

```
#!/bin/sh
# /etc/ppp/ip-up
/etc/rc.d/ipnat forcestart
```

```
#!/bin/sh
# /etc/ppp/ip-down
/etc/rc.d/ipnat forcestop
```

Diese Skripte werden vom *pppd* automatisch beim Eintreten der entsprechenden Ereignisse ausgeführt. Nur noch die Skripte kurz ausführbar machen und fertig ist das ppp-Gateway:

```
# chmod u+x /etc/ppp/ip-up /etc/ppp/ip-down
```

8.2.1 Ideen zum weiteren Ausbau

Man kann alles oben Beschriebene noch verbessern und besonderen Gegebenheiten anpassen. Hier nur ein paar Beispiele:

1. Um die Anzahlen der Einwahlen zu reduzieren, ist zum Beispiel ein Proxy interessant. Diese Proxies speichern Webseite zwischen und halten sie eine Weile, eine Einwahl ist somit nicht mehr nötig, wenn die Seite in letzter Zeit schon mal angefordert wurde. Dafür gibt es einige Lösungen in den Packages, beispielsweise *squid*.
2. Ein Caching Name Server kann ebenfalls Anfragen an das Internet reduzieren.

3. Wenn der Rechner schon mal online ist, warum dann nicht automatisch Mail und News austauschen? Dies kann in den *ip-up*- und *ip-down*-Skripten leicht implementiert werden.
4. Bisher liegen die Services auf dem Router recht offen. Theoretisch kann jeder von außerhalb sich mit dem Rechner verbinden. Aus Sicherheitsgründen sollte der IP-Verkehr noch gefiltert werden (mit einer Firewall).

8.3 Mit ISDN ins Internet

Eine modernere Technik, sich ans Internet anzubinden, ist ISDN. Angeboten werden zwei verschiedene Gerätetypen: Terminaladapter und ISDN-Karten. Diese Terminaladapter verhalten sich wie Modems und sind genauso zu konfigurieren. Diese werden genauso wie vorher beschrieben gehandhabt. Im folgenden wird also die Rede von ISDN-Karten sein.

Anmerkung

ISDN-Karten verwenden eine Variante von ppp, das synchrone ppp.

8.3.1 Die Kernelkonfiguration für ISDN

Die Konfiguration für einen Kernel mit ISDN-Unterstützung ist eigentlich recht einfach. In die Kernelkonfigurationsdatei sind der Treiber für die Karte und das ISDN-System einzutragen. An dieser Stelle wird nicht auf alle möglichen Karten und Treiber eingegangen. Hierzu sind Erklärungen in den Beispielkonfigurationsdateien für Kernel und in den Manpages für die Karten zu finden. Trotzdem liste ich zu Demonstrationszwecken die notwendigen Einträge am Beispiel einer IDSN-Karte Teles S0/16 auf:

```
# Teles S0/16 oder Creatix ISDN-S0 oder Niccy 1016
options         ISICISA_TEL_S0_16
isic0 at isa? port 0xd80 iomem 0xd0000 irq 9

#-----------------------------------------------------------------
# Ab hier das ISDN System
pseudo-device   isdn          # Kommunikation zwischen Kernel und isdnd
pseudo-device   isdntrc     2 # Geräte zum Debuggen der Verbindung
pseudo-device   isdnctl       # Gerät zur Steuerung der Verbindung
pseudo-device   isdnbchan   4 # Zugriff auf den B-Kanal
pseudo-device   isdntel     2 # Telefongerät
options         IPR_VJ        # Van Jacobsen Kompression
pseudo-device   irip        2 # Netzwerktreiber für IP über HDLC
                              # ISDNnetwork
pseudo-device   ippp        2 # synchrones PPP
```

Neben der im folgenden beschrieben Vorgehensweise kann man den *isdnd* auch noch über ein Programm namens *isdnmonitor*(8) steuern. Dafür wird das Gerät *isdnctl* benötigt. Das Gerät *isdntel* agiert als Telefon oder besser: als Anrufbeantworter. Da hier aber eine Menge Arbeit steckt, sollte sich der Leser damit in aller Ruhe selbst beschäftigen. Schließlich geht es hier um die Verbindung zum Internet.

Anmerkungen

Auf dem Amiga wird nur eine ISDN-Karte unterstützt, daher ist dort die Kernelkonfiguration recht einfach:

```
aster* at zbus?
isic* at aster? port?
```

Dies ist auch ein gutes Beispiel für die Architektur von NetBSD. Der isic-Treiber wurde schon weiter oben auf dem i386 (PC) verwendet.

8.3.2 Konfiguration des Zugangs

Die Basiskonfiguration der »/etc/hosts«, »/etc/nsswitch« und »/etc/resolv.conf« ist die gleiche wie im Abschnitt über ppp geschrieben.

Ähnlich wie beim analogen Modem werden die Konfigurationsdateien für ISDN im Verzeichnis »/etc/isdn« gesammelt. Es muß vorher angelegt werden:

```
# mkdir /etc/isdn
```

Als nächstes sollen die Dateien »holidays« und »isdnd.rates« in diesem Verzeichnis installiert werden. Die Datei »holidays« nimmt die Daten von Tagen auf, die als Feiertag gelten. »isdn.rates« beschreibt die normalen Verbindungstarife. NetBSD stellt dafür Beispiele im Verzeichnis »/usr/share/examples/isdn« zur Verfügung. In diesem Verzeichnis besitzen die Dateien etwas unterschiedliche Namen, beispielsweise enthält »isdn.rates.D« Tarifinformationen für die Deutsche Telekom. Sind passende Dateien vorhanden, kann man sie einfach kopieren. Ansonsten können die Dateien nach den Vorlagen angepaßt werden. Man kann auch leere Dateien nehmen und verzichtet damit auf ein wenig Komfort. Für den Anfang reichen leere Dateien aus.

Um die Konfiguration überschaubar zu halten, wird der Standardfall, die einzelne Verbindung zum Provider, beschrieben. Diese setzt das Gerät *ipppp0* (ISDN ppp) ein. Es ist nur eine Konfigurationsdatei für den *isdnd* (dem Gegenstück zum *pppd*) zu schreiben:

```
system
ratesfile          = /etc/isdn/isdnd.rates # Datei mit Tarifinformationen
```

```
entry
name                    = world  # Name für diese Verbindung
usrdevicename           = ippp
usrdeviceunit           = 0
ppp-auth-paranoid       = no
ppp-send-auth           = pap                   # oder "chap"
ppp-send-name           = "weltzugang"          # Benutzerkennung
ppp-send-password       = "Dkj143"              # Passwort
isdncontroller          = 0
isdnchannel             = -1
direction               = out
local-phone-dialout     = XXXXX    # Die eigene Telefonnummer
remote-phone-dialout    = XXXXX    # Die Telefonnummer des Providers
remdial-handling        = first
dialin-reaction         = reject
dialout-type            = normal
b1protocol              = hdlc
unitlength              = 0
unitlengthsrc           = conf
idletime-outgoing       = 1000
earlyhangup             = 0
dialretries             = 5
dialrandincr            = on
recoverytime            = 2
usedown                 = off
downtries               = 2
downtime                = 30
```

Dies ist eine Beispielkonfiguration für Flatrates. Lediglich ein paar Einträge müssen für den Provider angepaßt werden:

ppp-send-auth	Dies kann *pap* oder *chap* sein. Die Information erhält man vom Provider oder man probiert selbst aus.
ppp-send-name	Hier muß die Benutzerkennung vom Provider eingetragen werden. Das ganze in Anführungsstriche zu setzen, ist eine gute Idee.
ppp-send-passwort	Hier wird das Paßwort eingetragen.
local-phone-dialout	Hier die eigene Telefonnummer eintragen. Möglicherweise interessiert es den Provider nicht, aber ein ISDN-Netzwerk (beispielsweise Telefonanlage in Unternehmen) kann Verbindungen von unregistrierten Nummern abbrechen.
remote-phone-dialout	Die Telefonnummer des Providers eintragen.

Als nächstes ist eine Datei »/etc/ifconfig.ippp0« anzulegen:

```
inet 0.0.0.0 0.0.0.1 link1 down
```

Dies konfiguriert das Interface *ippp0* mit den folgenden Einstellungen:

1. Die lokale Adresse wird auf 0.0.0.0 gesetzt. Hierdurch muß die Gegenseite die IP-Nummer für den Rechner liefern.
2. Die Adresse der Gegenseite wird auf 0.0.0.1 gesetzt. Die Gegenseite muß die eigene IP-Nummer mitteilen. Ist die IP-Nummer vom Provider bekannt, sollte diese hier eintragen werden.
3. Der Zusatz *link1* bedeutet, daß die Verbindung bei Bedarf aufgebaut wird.
4. Das Interface wird nicht als *up* markiert. Da das Interface beim Booten recht früh konfiguriert wird – vor dem Start des *isdnd* – werden damit Fehlversuche vermieden. Der *isdnd* wird später das Interface aktivieren.

Jetzt muß die Verbindung in die Routingtabellen aufgenommen werden. In diesem Beispiel wird von einer einzigen Verbindung, nämlich ISDN, ausgegangen. In der Datei »/etc/defaults/rc.conf« wird der Eintrag *defaultroute* geändert:

```
defaultroute="0.0.0.1 -iface"
```

Ist die IP-Nummer des Einwahlpunkts bekannt, sollte sie hier auch wieder eingetragen werden. Da die »rc.conf« sowieso geändert wird, kann man auch noch einen weiteren Eintrag tätigen:

```
isdnd=YES
```

Nach dem Neustartdes Rechners sollte die ISDN-Verbindung aktiv sein.

8.3.3 ISDN bei Bedarf

Bisher wurde eine permanente Verbindung beschrieben, die während des Bootens aktiviert wird. Eine automatische Trennung erfolgt nicht. Dies soll nun abgestellt werden. In die »isdnd.rc« ist folgende Zeile einzutragen, damit das Interface nicht vom *isdnd* sofort aktiviert wird:

```
autoupdown=no
```

Und für das automatische Trennen der Verbindung benötigt man noch die folgenden Zeilen:

```
idletime-incoming     = 120  # Eingehende Verbindung nach 120
                             # Sekunden Inaktivität trennen
idletime-outgoing     = 55   # Ausgehende Verbindung nach 55
                             # Sekunden Inaktivität trennen
earlyhangup           = 5    # Verbindung 5 Sekunden vor der
                             # Einheit trennen
ratetype              = 0    # Eintrag aus den Tariftabellen
unitlength            = 90   # Einheitenlänge in Sekunden
unitlengthsrc         = aocd # Gebühreninformationen werden
                             # übermittelt
```

Die Konfiguration der automatischen Trennung kann noch stark erweitert werden. Hierzu ist das Studium von *isdnd*(8) zu empfehlen.

Die Konfiguration für die Anbindung mehrerer Rechner folgt analog den Anweisungen im Abschnitt über ppp.

8.4 ppp über das Ethernet – ADSL-Verbindungen

ADSL ist zur Zeit das einzige Breitbandverfahren für eine WAN-Anbindung, das für Privatpersonen preislich akzeptabel ist. Genauer betrachtet geschieht die Konfiguration von ADSL auch nicht viel anders als bei einem normalen Modem. Verwendet wird das Protokoll PPP over Ethernet (*pppoe*). Die Konfiguration geschieht ähnlich der des analogen Modems, allerdings wird nicht die serielle Leitung, sondern ein Ethernetkabel mit dem Modem verbunden.

8.5 Kernelkonfiguration

In den Kernel (für alle Plattformen) ist das zuständige für *pppoe*-Gerät aufzunehmen:

```
pseudo-device    pppoe
```

Der manuelle Test der Verbindung ist recht einfach durchzuführen. Was muß gemacht werden? Das Interface *pppoe0* wird konfiguriert:

```
# ifconfig pppoe0 create
# ifconfig pppoe0 inet 0.0.0.0 0.0.0.1 down
```

Die IP-Nummer wird später durch den Provider zugeteilt. Deshalb sind hier 0.0.0.0 beziehungsweise 0.0.0.1 eingetragen.

Das Interface *pppoe0* wird zunächst noch nicht aktiviert. Zur Zeit ist dem Gerät ja noch kein Ethernet-Interface zugewiesen, weswegen das Modem auch nicht angesprochen werden kann. Das erfolgt jetzt. Als Ethernet-Interface wird im Beispiel *ne0* verwendet, jedes andere geht aber auch:

```
# ifconfig ne0 up
# pppoectl -e ne0 pppoe0
```

Wie bei allen bisher beschriebenen Anbindungen wird die Authentifizierung auch noch konfiguriert:

```
# pppoectl pppoe0 myauthproto=pap 'myauthname=weltzugang' \
  'myauthsecret=Dkj143' hisauthproto=none
```

Die Angaben wie *myauthproto*, *myauthname* und *myauthsecret* sind anzupassen

Jetzt ist alles für die erste Verbindung vorbereitet. Falls sich bei der Konfiguration noch ein Fehler eingeschlichen haben sollte, wird vorher noch die automatische Wiederholung abgeschaltet:

```
# pppoectl pppeo0 max-auth-failure=1
```

Wenn man jetzt das Interface *pppoe0* aktiviert, sollte die Verbindung aufgebaut werden:

```
# ifconfig pppoe0 up
```

Zur Kontrolle fragt man *pppoe0* nach seinem Zustand:

```
# pppoectl -d pppoe0
pppoe0: state = session
        Session ID: 0x1234
        PADI retries: 0
        PADR retries: 0
```

Steht in der Ausgabe wie oben *state = session*, funktioniert alles. Andernfalls geht es an die Fehlersuche. Beispielsweise könnten Autorisierungsdaten falsch eingegeben sein. Die ADSL-Konfiguration selbst ist vielleicht ausreichend.

Ein solcher Fehler wird durch die Felder *PADI* und *PADR* angezeigt. Unter Umständen muß ein Service-Name (Option *-s* für *pppoectl*) oder der Name eines Zugangspunkts (engl. Access Concentrator) (Option *-a* für *pppoectl*) angegeben werden. Die Provider sollten diese Informationen dokumentiert haben und bereitstellen.

Funktioniert die Verbindung schon, ist es an der Zeit, die IP-Nummer zu überprüfen:

```
# ifconfig pppoe0
pppe0: flags=8851 mtu 1492
       inet 123.123.123.123 -> 123.1.1.1 netmask 0xff000000
```

Man sieht zwei IP-Nummern – die lokale 123.123.123.123 und die IP-Nummer 123.1.1.1 des Zugangs. Sollte dies bei Ihnen nicht so ausschauen, ist die Authentifizierung nochmals zu prüfen.

8.5.1 Permanente Verbindungen (Flatrate)

Die oben erfolgreich durchgeführte manuelle Konfiguration wird jetzt festgehalten, damit der Rechner nach dem Systemstart automatisch die Verbindung zum Internet herstellt. Dies geschieht durch einen Eintrag in der Datei »/etc/ifconfig.pppoe0«:

```
create
# Ethernet-Interface zuweisen
! /sbin/ifconfig ne0 up
! /sbin/pppoectl -e ne0 pppoe0
# Authentifizierung konfigurieren
! /sbin/pppoectl pppoe0 myauthproto=pap "myauthname=weltzugang" \
"myauthsecret=Dkj143" hisauthproto=none
# und das Interface selbst
# alle IP-Nummern werden zugewiesen
0.0.0.0 0.0.0.1 up
```

Aus Sicherheitsgründen sollte diese Datei nur für *root* lesbar sein.

Nun zum Rest der Konfiguration, zu den Skripten *ip-up* und *ip-down*:

```
#!/bin/sh
# /etc/ppp/ip-up
/sbin/route add default $5
```

und

```
#!/bin/sh
# /etc/ppp/ip-down
/sbin/route add default $5
```

Beide Skripte brauchen das Ausführungsrecht für *root*.

Zuletzt ist noch ein Eintrag in die »/etc/defaults/rc.conf« vorzunehmen:

```
ifwatchd=YES
```

Durch diesen Eintrag in der »/etc/defaults/rc.conf« wird beim Systemstart ein Daemon *ifwatchd*(8) gestartet, der die oben beschriebenen *ip-up*- und *ip-down*-Skripte startet. Der Start dieses Daemons erfolgt aber im Vergleich zur Interfacekonfiguration recht spät, dabei können in der Zwischenzeit noch andere Netzwerkservices, beispielsweise *named* oder *ntpd* gestartet werden.

Da erst im *ip-up*-Skript eine Standardroute für das Netzwerk konfiguriert wird, sollte man diese Services im *ip-up*-Skript restarten.

8.5.2 Einwahl bei Bedarf

Bei der Einwahl bei Bedarf unterscheidet sich die Konfiguration in drei Punkten von der der Flatrate:

1. Das Flag *link1* wird dem Interface *pppoe0* bei der Konfiguration mitgegeben.
2. Der Leitung soll nach einer bestimmten Zeit der Nichtbenutzung abgebaut werden.
3. Die Route über das Interface *pppeo0* muß auch bestehen, wenn die Verbindung nicht aufgebaut ist. Nur so kann bei Bedarf die Einwahl erfolgen.

Die geänderte »/etc/ifconfig.pppoe0« sieht jetzt so aus:

```
create
# Ethernetinterface zuweisen
! /sbin/ifconfig ne0 up
! /sbin/pppoectl -e ne0 pppoe0
# Authentifizierung konfigurieren
# Abbau der Leitung nach 60 Sekunden
! /sbin/pppoectl pppoe0 idle-timeout=60 myauthproto=pap \
'myauthname=weltzugang' 'myauthsecret=Dkjl43' hisauthproto=none
# und das Interface selbst
# alle IP-Nummern werden zugewiesen
0.0.0.0 0.0.0.1 link1 up
! /sbin/route add default -iface 0.0.0.1
```

Der Start des Daemons *ifwatch*(8) ist in dieser Konfiguration nicht erforderlich, er kann aber trotzdem gestartet werden wie dies bereits im Abschnitt über DSL-Flatrate beschrieben wurde. Als Anwendungsmöglichkeit hierfür sei der Mail- und Newsaustausch erwähnt.

Im folgenden ein paar Anmerkungen zu den Skripten *ip-up* und *ip-down*: Der *pppd*(8) kann in den verschiedenen Phasen der Verbindung unterschiedliche Skripte aufrufen, damit auf verschiedene Situationen reagiert werden kann. Für alle diese Skripte setzt der *pppd* einige Umgebungsvariablen und übergibt ihnen zusätzlich noch ein paar Argumente. Der Daemon *ifwatchd*(8) übernimmt die Aufgabe des Starts dieser Skripte für Verbindungen – beispielsweise für ADSL –, die nicht über den *pppd* abgewickelt werden.

Variable	Funktion
DEVICE	Name der benutzten seriellen Schnittstelle.
IFNAME	Name des benutzten Netzwerk-Interfaces.
IPLOCAL	Die lokale IP-Nummer.
IPREMOTE	Die IP-Nummer der Gegenstelle.
PEERNAME	Der Authentifizierungsname der Gegenstelle.
SPEED	Geschwindigkeit der seriellen Schnittstelle.
ORIG_UID	Die Nummer des Benutzers, der pppd aufrief.
PPPLOGNAME	Benutzername zu *ORIG_UID*.

Tabelle 8.1: Liste der Umgebungsvariablen

Für die *ip-down*- und *auth-down*-Skripte werden zusätzlich gesetzt:

Variable	Funktion
CONNECT_TIME	Zeit seit Aufbau der Verbindung in Sekunden.
BYTES_SENT	Anzahl der verschickten Byte.
BYTES_RCVD	Anzahl der empfangenen Byte.
LINKNAME	Logischer Name der Verbindung.

Tabelle 8.2: Liste der zusätzlichen Umgebungsvariablen

Zwei der wichtigsten Skripte sind *ip-up* und *ip-down*, wie sie oben schon mehrfach verwendet wurden.

Das Skript »/etc/ppp/ip-up« wird aufgerufen, nachdem die Verbindung bereit ist, IP zu übertragen. Das Skript erhält als Argumente:

1. Den Namen des Netzwerkinterfaces.
2. Den Namen der seriellen Schnittstelle.
3. Die Geschwindigkeit der seriellen Schnittstelle.
4. Die lokale IP-Nummer.
5. Die IP-Nummer der Gegenstelle.
6. Der *pppd* kennt ein Argument *ipparam*. Das Argument, das nach dem *ipparam* beim Aufruf von *pppd* folgt, wird hier als Argument eingesetzt.

Dieses Skript ermöglicht durch diese Informationen Netzwerkeinstellungen zu vervollständigen, die vorher noch nicht bekannt waren.

Das Skript */etc/ppp/ip-down* wird aufgerufen, sobald kein IP-Verkehr mehr über die Leitung stattfinden kann. Es bekommt die gleichen Argumente wie *ip-up* übergeben und macht Netzwerkeinstellungen durch *ip-up* wieder rückgängig.

Neben diesen Skripten für IPv4 sind Skripte mit gleicher Funktionalität für die Protokolle IPX und IPv6 definiert. Da sie zur Zeit selten verwendet werden, sollte die Erwähnung ausreichen.

8.6 WaveLAN – drahtlos verbinden

Eigentlich gehört dieses Thema nicht ganz hierher, da drahtlose Vernetzung von Rechnern noch relativ neu und relativ unbekannt ist.

Da sich aber jeder in den Funkverkehr einschalten kann, sollte auf Sicherheit größter Wert gelegt werden. Schließlich will man ja keine fremden Rechner in seinem eigenen Netz haben. Vielleicht ist es eine gute Idee, sich in diesem Zusammenhang mit *IPsec* auseinander zusetzen?

8.6.1 Kernelkonfiguration

Wie üblich sind ein paar Gegebenheiten in den Kernel, hier für Lucent WaveLAN, ELSA AirLancer MC-11 einzubauen:

```
wi*     at pcmcia?   function ?
```

Auch für die Steckkartenversion dieser Geräte ist der PCMCIA-Bus zu verwenden. Die Karten besitzen dafür den Interfacebaustein. Für den MacPPC-Port gilt ähnliches:

```
wi*     at obio?        # AirMac
```

beziehungsweise

```
wi*     at pcmcia? function?     # Lucent WaveLan
```

8.6.2 Setup fürs Netzwerk

Nach dem Booten des neuen Kernels sollte die Karte gefunden worden sein und es geht los mit der Konfiguration. Hierfür werden die Befehle *wiconfig*(8) und *ifconfig*(8) verwendet. Als erstes wird die Konfiguration der Karte ausgelesen:

```
# wiconfig -i wi0
NIC serial number:                  [ 98UT12345678 ]
Station name:                       [ NetBSD WaveLAN/IEEE node ]
SSID for IBSS creation:             [ NetBSD IBSS ]
Current netname (SSID):             [ NetBSD IBSS ]
Desired netname (SSID):             [ ]
```

```
Current BSSID:                    [ 00:00:00:00:00:00 ]
Channel list:                     [ 2047 ]
IBSS channel:                     [ 3 ]
Current channel:                  [ 3 ]
Comms quality/signal/noise:       [ 0 27 27 ]
Promiscuous mode:                 [ Off ]
Port type (1=BSS, 3=ad-hoc):      [ 3 ]
MAC address:                      [ 00:7f:de:12:34:56 ]
TX rate:                          [ 3 ]
RTS/CTS handshake threshold:      [ 2347 ]
Create IBSS:                      [ Off ]
Access point density:             [ 1 ]
Power Mgmt (1=on, 0=off):         [ 0 ]
Max sleep time:                   [ 100 ]
```

Normalerweise wird der »Port type« angepaßt. Hier ist meist auf *ad-hoc* eingestellt, was zum Testen und für Verbindungen zwischen zwei Geräten ausreicht. Häufig wird man dies aber auf BSS (Basic Service Set) umstellen. In dieser Betriebsart kann man mit den Access Points, der Verbindung zwischen drahtlosem und drahtgebundenem Netz, kommunizieren. Dafür verwendet man *wiconfig*(8):

```
# wiconfig wi0 -p 1
```

Der Name des Netzwerks wird gesetzt:

```
# wiconfig wi0 -n meinnetz
```

Jetzt noch den Kanal *Channel* der Karte anpassen. Leider sind nicht alle Kanäle in den unterschiedlichen Ländern legal. Hierzu ist in der Dokumentation des Gerätes nachzusehen. Jedenfalls müssen die Karte und der Access Point auf dem gleichen Kanal arbeiten:

```
# wiconfig wi0 -x 1
```

Der Rest der Netzwerkkonfiguration funktioniert genauso wie beim Ethernet.

Anmerkung

Für die Apple Airport Base Station gibt es in den Packages ein Javaprogramm *airportbasestationconfig*. Damit kann die Station einfach konfiguriert werden.

8.6.3 Erstellen der Konfigurationsdateien

Um die gemachten Einstellungen permanent festzuhalten, ist noch die Datei »/etc/ifconfig.wi0« anzulegen:

```
!/usr/sbin/wiconfig wi0 -n meinnetz
!/usr/sbin/wiconfig -p 1
!/usr/sbin/wiconfig -c 1
inet 192.168.1.32 netmask 255.255.255.0
```

Erläuterung:

1. Der Netzwerkname wird gesetzt.
2. Der Port Type wird angepaßt.
3. Der Kanal ausgewählt.
4. Dies ist das Argument für Aufruf von *ifconfig* bei Systemstart.

Die einzelnen Einstellungen wie IP-Nummer und Netzmaske und so weiter müssen natürlich angepaßt werden.

Bei Problemen wird auf die Manpage *wiconfig*(8) und *ifconfig*(8) verwiesen.

8.7 VPN (Virtual Private Network) mit ppp über SSH

Sicherheit ist heutzutage ein vielschichtiges Thema und die gesicherte »Leitung« zwischen zwei Rechnern zählt dazu. Eine Möglichkeit, so etwas zu realisieren, ist die Verschlüsselung der ppp-Verbindung.

Wie soll dies technisch funktionieren?

In der Vergangenheit wurde eine Kombination von Tastatur und Drucker, ein Fernschreibterminal (engl. Teletyper), an die serielle Schnittstellen angeschlossen und ermöglichte auf diesem Weg die Kommunikation mit dem Rechner. Die Druckwerke wurden im Laufe der Jahre durch Bildschirme ersetzt. Bei langen Leitungswegen wurden irgendwann Modems eingesetzt. Mit dem Aufkommen des Netzwerks wurde diese alte Terminalschnittstelle auf das Netzwerk übertragen, das war das Entstehen der Pseudoterminals. Aus Sicht der Software ist es egal, ob man an einem realen Terminal oder über das Netz an einem Pseudoterminal sitzt. Die Grundidee von ppp über SSH ist, eine Terminalverbindung zu einem anderen Rechner mit Hilfe von SSH aufzubauen und diese für ppp zu nutzen.

Im folgenden wird in erster Linie das Prinzip beschrieben. Weitere Verfeinerungen und Verbesserungen sind in vielen Punkten sinnvoll und notwendig.

Um zwei Rechner mit einer ppp-Verbindung auszustatten, werden zwei IP-Nummern benötigt, beispielsweise 192.168.1.16 und 192.168.1.32. Über diese Adressen soll die gesicherte Verbindung durchgeführt werden. Hier wurden die Nummern aus einem für das Internet gesperrten Bereich gewählt. Diese Nummern haben den Vorteil, zur Verfügung zu stehen und den Nachteil, nicht ins Internet zu gehören. Da aber nur eine

Verbindung zwischen zwei Rechnern geplant ist, soll dies nicht weiter kümmern. Für Verbindungen zwischen Netzwerken ist IPNAT bei solchen IP-Nummern Pflicht.

Eine der IP-Nummern (192.168.1.16) weist man den Server zu, die andere (192.168.1.32) dem Client. Für diese Nummern kann man wieder Einträge in »/etc/hosts« vornehmen.

8.7.1 Die SSH-Verbindung vorbereiten

Als erstes wird ein neuer Benutzer namens *vpn* mit der Gruppe *vpn* angelegt.

```
# groupadd vpn
# useradd -m -g vpn -G wheel -c "VPN Pseudo User" vpn
```

Die Verbindung soll mit Hilfe von SSH erfolgen und schließlich will man sich nicht jedes Mal komplett mit Paßwort einloggen. Es ist also ein Schlüssel (als Benutzer *root*) anzulegen:

```
# ssh-keygen
```

Dieses Kommando erzeugt ein Paar von Schlüsseln, einen öffentlichen und einen privaten. Der öffentliche Schlüssel wird später zur Authentifizierung verwendet.

Jetzt wird die Datei »identity.pub« auf dem Server in die Datei »authorized_keys« des Benutzers *vpn* kopiert. Dies ist von Hand zu erledigen. Zuerst ein paar vorbereitende Schritte:

```
# cd ~/vpn
# mkdir .ssh
# chown root:vpn .ssh
# chmod 755 .ssh
# cd .ssh
```

Nun kann die Datei kopiert werden:

```
# cp /root/.ssh/identity.pub authorized_keys
# chown root:vpn authorized_keys
# chmod 644 authorized_keys
```

Die Datei »authorized_keys« besteht aus einer einzigen Zeile. Weitere Zeilen können an dieser Datei angefügt werden, damit sich jeder Client authentifizieren kann. Diese Prozedur ist also auf den Client nochmals durchzuführen, die so erzeugte »identity.pub« ist auf den Server zu übertragen und in die »authorized_keys« aufzunehmen.

Bisher kann sich ein Benutzer des Client auf dem Server als *vpn* einloggen. Leider kann *vpn* immer noch nicht den *pppd*. Dafür muß er erst einen Benutzerwechsel durchführen,

wofür er aber das Paßwort von *root* braucht. Um dies zu umgehen, wird *sudo* aus den Packages installiert. In die Konfigurationsdatei »/usr/pkg/etc/sudoer« werden folgende Zeilen eingetragen:

```
Cmnd_Alias VPN=/usr/sbin/pppd
slave ALL=NOPASSWD: VPN
```

Mit dem Kommando

```
pppd updetach noauth passive pty \
     "ssh -P vpn.server.de -L vpn -o Batchmode=yes \
/usr/pkg/bin/sudo -c /usr/sbin/pppd nodetech notty noauth" \
     ipparam vpn 192.168.1.32:192.168.1.16
```

wird jetzt die Verbindung aufgebaut. Um das VPN einfacher einsetzen zu können, sollte das Kommando als Skript angelegt werden. Ein solches Skript kann auch die Startup-Skripte im Verzeichnis »/etc/rc.d« aufgenommen werden.

Zur Erläuterung hier die Übersicht der verwendeten Parameter:

pppd	Der erste Aufruf von *pppd* ist der Endpunkt auf der Seite des Clients.
updetach	Der *pppd* soll das Terminal, von dem aus er gestartet wird, nach dem Aufbau der Verbindung wieder freigeben.
noauth	Es erfolgt keine Authentifizierung über den *pppd*. Dies wird von SSH übernommen.
passive	Der *pppd* wird sich passiv verhalten. Er soll keine IP-Nummern bei der Gegenseite anfragen.
pty	Das folgende Argument wird als Kommando aufgefaßt und auf der Clientseite gestartet: ♦ *ssh*: Die SSH-Verbindung zum Rechner wird aufgebaut. ♦ *-P*: Die SSH-Kommunikation wird über nicht privilegierte Ports durchgeführt. Diese Option ist nicht unbedingt notwendig und dient nur zum Umgehen einer eventuell vorhandenen Firewall. ♦ *vpn.server.de*: Der Rechner, der als Server für die VPN-Verbindung dienen soll. Die IP-Nummer dieses Rechners gehört zum VPN. ♦ *-L*: Eingeloggt wird als Benutzer *vpn*. ♦ *-o Batchmode=yes*: Die Gegenseite soll im Batchmode laufen.

	♦ *sudo -c pppd nodetach notty noauth*: Start des *pppd* auf dem Server: ♦ *su -c*: Das folgende Argument wird als *root* ausgeführt. ♦ *pppd*: Start des *pppd* auf dem Server. ♦ *nodetach*: siehe oben. ♦ *notty*: Diese Verbindung belegt kein Terminal. ♦ *noauth*: Siehe oben. Beim Start dieses Kommandos erfolgte die Authentifizierung bereits durch SSH.
ipparm	Das folgende Argument, hier *vpn*, wird an das Skript *ip-up* als Argument übergeben.
192.168.1.32: 192.168.1.16	Die IP-Adresse der Verbindung. Die Clientseite wird hier keine IP-Nummer anfordern (siehe *passive*). Die Serverseite wird dies übernehmen und die Clientseite muß antworten. Somit sollte der Client die Daten der Verbindung haben.

9 Datenaustausch im Netz

9.1 Datenaustausch via NFS

von Dr. Lex Wennmacher

Mit Hilfe von NFS ist es möglich, ein gesamtes Dateisystem oder einen Unterverzeichnisbaum daraus (auf dem sogenannten NFS-Server) über das Netzwerk einem anderen Rechner (dem sogenannten NFS-Client) zur Verfügung zu stellen. Der Client kann dabei (anders als zum Beispiel beim Datenaustausch über ftp) mit den Standard-Datei-Systemaufrufen *open*(2), *read*(2) und *write*(2) auf die Dateien zugreifen, also ganz so, als ob die Daten lokal auf dem Rechner vorhanden wären.

NFS wurde wie viele andere heute benutzten Netzwerk-Standards im Jahr 1985 von der Firma Sun Microsystems entwickelt. Sun entwarf dabei nicht nur eine NFS-Implementation, sondern publizierte auch den dazugehörigen Standard. Aus einer Reihe von Gründen – unter anderem lizenzrechtlicher Natur – konnte Berkeley die Original-Sun-Implementation nicht übernehmen. Man entschloß sich stattdessen, NFS zu reimplementieren. Dabei hielt man sich nicht in allen Aspekten an den von Sun vorgegebenen NFS-Standard, sondern verbesserte ihn in einigen Punkten (zum Beispiel bei den sogenannten »Leases«). Die daraus resultierende Implementation erhielt daher den Namen *nqnfs* (not quite NFS, nicht ganz NFS). Das in NetBSD verwendete nqnfs ist völlig abwärtskompatibel zu NFS Version 2, viele der Verbesserungen des nqnfs-Protokolls sind in NFS Version 3 eingeflossen.

Wie sieht nun das Grundprinzip der Funktionsweise von NFS aus? Wichtig zu wissen ist zunächst, daß es sich bei NFS um ein zustandsloses Protokoll handelt. Das bedeutet, daß sich der NFS-Server nicht den Zustand eines NFS-Clients merkt. Alle NFS-Operationen werden als Fern-Prozeduraufrufe (Remote Procedure Calls, RPC) abgewikkelt. Der Server prüft jeweils nur, ob der RPC-Aufruf von einer berechtigten Maschine stammt und führt ihn aus.

Fern-Prozeduraufrufe sind vergleichbar mit lokalen Prozeduraufrufen. Zunächst werden die Argumente für den RPC gemarshalled, das heißt, zusammengebündelt in einen Puffer gelegt, in ein bestimmtes Transportformat umgewandelt und über das Netzwerk einem entfernten Rechner übermittelt. Dieser führt nun den umgekehrten Prozeß durch (Zurückwandeln aus dem Netzwerk-Format in das Binärformat entsprechend der eigenen Maschinenarchitektur und Zerlegen in die einzelnen Argumentbausteine). Dann führt der entfernte Rechner mit diesen Argumenten einen Prozeduraufruf durch. Dessen Ergebnisse werden auf gleiche Weise wieder zurückübertragen. Bis zum Eintreffen des Ergebnisses wartet der aufrufende Rechner (beziehungsweise arbeitet an einem anderen Prozeß). Die zurückgelieferten Daten werden wieder in das eigene Maschinenformat umgewandelt und als Ergebnisse an den Aufrufer zurückgegeben. Programmierer finden die notwendigen Details zu RPC in der Manual-Seite *rpc*(3).

Damit Rechner mit unterschiedlichen Maschinenformaten (Little Endian und Big Endian) über das Netzwerk miteinander Daten austauschen können, hat Sun Microsystems ebenfalls ein spezielles Datenformat hierfür definiert: XDR (eXternal Data Representa-

tion). Grob gesagt legt XDR für jeden Maschinentyp exakt fest, in welcher Reihenfolge jedes einzelne Bit über die Leitung geht. NetBSD stellt, wie jedes andere gute Unix auch, als Teil seiner RPC-Implementation XDR-Routinen in der *libc* zur Verfügung. Wer mehr über XDR wissen will, sei auf die Manual-Seite *xdr*(3) hingewiesen.

Um größtmögliche Flexibilität zu gewährleisten, werden für RPCs keine festen Ports oder Adressen vergeben, sondern es wird eine Art Übersetzer verwendet. Wenn ein Client also einen RPC durchführen will, kontaktiert er zuvor diesen Übersetzer (auf dem entfernten Rechner) mit dem Namen *rpcbind*, um sich die Adresse geben zu lassen, an die der nachfolgende RPC dann geschickt wird. Damit *rpcbind* diese Übersetzungen alle durchführen kann, muß *rpcbind* auf jeder Maschine, die RPC-Anfragen annehmen soll, als erster RPC-Dienst gestartet werden. Alle später gestarteten RPC-Dienste registrieren ihre Adressen und die Dienste, die sie bedienen, beim *rpcbind*, so daß dieser dann die Anfragen der Netzwerk-Clients beantworten kann. Man erkennt übrigens, daß wegen dieses Registrierungsmechanismus auch alle RPC-Dienste neu gestartet werden müssen, falls einmal der Neustart des *rpcbind* erforderlich werden sollte.

RPC als solches kann sowohl über TCP als auch über UDP betrieben werden. Die NetBSD-Implementation des NFS-Servers unterstützt – anders als viele andere Unix-Systeme (beispielsweise Linux) – sowohl TCP/IP als auch UDP/IP als zugrunde liegendes Transportprotokoll. Die Unterstützung von TCP ist ein enormer Vorteil, wenn NFS-Mounts über Netze vorgenommen werden sollen, bei dem man in höherem Maße mit Paketverlusten rechnen muß.

9.1.1 Vorbereitende Arbeiten

Zunächst muß natürlich auf beiden Rechnern, dem Server und dem Client, das Netzwerk korrekt aufgesetzt sein und die Netzwerk-Verbindung zwischen beiden Rechnern gewährleistet sein.

Sollten die Rechner beispielsweise durch eine Firewall getrennt sein, ist dafür Sorge zu tragen, daß Port 111 (für den RPC-Service) und Port 2049 (für den *nfsd*) geöffnet sind.

Je nach Anwendung kann auch die Synchronisation der Uhren von Server und Client erforderlich werden. Zu der Frage, wann die Uhrensynchronisation wirklich erforderlich wird, findet man in der Literatur oft nur unzureichende und zum Teil auch falsche Informationen. Hier soll deswegen dieser Themenkomplex einmal genauer besprochen werden.

Bei einem Schreibzugriff über NFS ruft letztlich der NFS-Server den Systemaufruf *write*(2) (oder einen ähnlichen Aufruf) auf. Als Teil dieses Aufrufs werden auch die entsprechenden Zeitstempel der Datei modifiziert, und zwar mit von der Uhr des NFS-Servers abgeleiteten Zeiten. Die Systemuhr des Client spielt für die Zeitstempel also keine Rolle. Selbst wenn viele Clients mit falsch gehenden Uhren quasi-parallel auf ein NFS-Dateisystem zugreifen, passiert nichts schlimmes, da die Zugriffe durch die Systemuhr des NFS-Servers serialisiert werden.

Ungemach droht lediglich dann, wenn ein NFS-Client den Zeitstempel einer NFS-Datei mit seiner eigenen Uhr vergleicht. Dann kann es sogar passieren, daß diese zur Verwirrung des Benutzers und des Systems aus der Zukunft zu stammen scheinen.

Ein Paradebeispiel für ein Unix-Kommando, bei dem Probleme mit Dateien aus der Zukunft zu erwarten sind, ist *make*. *make* wird benutzt, um eine Zieldatei nach bestimmten Regeln automatisch aus Quelldateien und möglichen Zwischendateien zu erstellen. Um seine Arbeit mit möglichst geringem Ressourcenverbrauch durchführen zu können, liest *make* ein Makefile, um herauszufinden, welche Datei von einer anderen abhängt und welche Schritte daraufhin überhaupt noch durchzuführen sind. Hierbei werden die Zeitstempel von Quell- und Zieldatei verglichen. Ist die Quelldatei neuer als die Zieldatei, muß die Zieldatei nach den im Makefile spezifizierten Regeln neu erstellt werden.

Dieses Vergleichen von Dateizeitstempeln ist über NFS völlig unkritisch, auch in dem Fall, wo eine oder beide Dateien in der Zukunft zu liegen scheinen. Der Haken beim Verwenden von *make* in einem NFS-Dateisystem liegt aber darin, daß es zu dateilosen Zwischenschritten kommen kann. In diesen Fällen muß *make* den Zeitstempel für den Zwischenschritt von der Systemuhr ableiten, womit ein mögliches Scheitern bereits vorprogrammiert ist.

Grundsätzlich ist also zur Vermeidung von Inkonsistenzen bei *make* eine Uhrensynchronisation zwischen NFS-Server und NFS-Client erstrebenswert. Ist das nicht möglich, da beispielsweise Server und Client in unterschiedlichen administrativen Domänen liegen, kann man versuchen, die Make-Datei so umzuschreiben, daß dateilose Zwischenschritte vermieden werden. Dies erreicht man dadurch, daß man alle Zwischenschritte grundsätzlich von (Dummy-) Dateien abhängig macht, deren Zeitstempel dann mit einem Aufruf von *touch* aktualisiert werden.

Zur Synchronisation von Uhren verweise ich auf Kapitel 7.6.3.

Die Rechte bei Zugriff über NFS werden genau wie beim Zugriff auf ein lokales FFS aus UID und GID bestimmt. NFS nimmt einfach an, daß die UIDs und GIDs auf Server und Client dieselben Benutzer beschreiben. Wer als Administrator also sicherstellen will, daß die Konsistenz der Zugriffsrechte auch über NFS gewahrt bleiben, muß sicherstellen, daß die Benutzer auf Server- und Client-Seite die gleichen UIDs und GIDs besitzen. Das kann natürlich manuell beim Einrichten der Accounts erfolgen oder besser durch Verwenden von netzwerkweiten Passwortsystemen, wie beispielsweise Yellow Pages (NIS). Die Konfiguration von Yellow Pages wird in Kapitel 7.6.5 beschrieben.

Ist es aus administrativen Gründen (zum Beispiel weil NFS-Server und -Client sich in unterschiedlichen administrativen Domänen befinden) nicht möglich, benutzereinheitliche UIDs und GIDs zu verwenden, kann man zwei verschiedene Tricks anwenden. Zum einen läßt NFS beim Mounten die Angabe eines General-Accounts zu, auf den alle UIDs abgebildet werden, die Details hierzu werden in 9.1.2 beschrieben. Zum anderen kann man eine UID-/GID-Abbildung durch eine Zwischenschicht des Dateisystems erwirken. Dieses Verfahren ist flexibler als die erste Methode, funktioniert zur Zeit allerdings auch nur bei bis zu sechzehn verschiedenen Benutzern und muß noch als »experimentell« eingestuft werden. Die Details werden in 9.1.4 beschrieben.

Eine Besonderheit stellt der NFS-Zugriff durch den Administrator *root* dar. Obwohl »Sicherheit« kein Designziel von NFS war, werden standardmäßig alle Zugriffe durch *root* auf der Client-Seite vom NFS-Server in Zugriffe durch den Benutzer *nobody* umgewandelt. Damit soll verhindert werden, daß die root-Privilegien auf dem NFS-Client unkontrolliert auf den NFS-Server übertragen werden.

9.1.2 NFS-Server einrichten

Das Einrichten eines NFS-Servers stellt den größten Teil der Arbeit dar, ist aber nicht wirklich kompliziert. Zuerst muß sichergestellt werden, daß der Kernel auf der NFS-Server-Maschine Unterstützung für die NFS-Server-Funktion enthält. Dies kann man zum Beispiel mit dem folgenden Befehl prüfen:

```
% nm -o /netbsd | grep nfssvc_addsock | wc -l
```

Wird eine 1 angezeigt, ist das Symbol *nfssvc_addsock* im Kernel »/netbsd« enthalten. Somit ist die NFS-Serverfunktion vorhanden. Falls eine 0 zurückgegeben wird, muß zuerst ein geeigneter Kernel gebaut werden. Das erreicht man dadurch, daß man in der Kernelkonfigurationsdatei der NFS-Servermaschine die Zeile

```
options         NFSSERVER           # Network File System server
```

aktiviert (indem man die Zeilen zur Konfigurationsdatei hinzufügt oder die vorhandenen führenden Kommentarzeichen entfernt) und den Kernel neu kompiliert. Die Details der Prozedur sind in Kapitel 4 beschrieben. Nach dem Kopieren des neuen Kernels nach »/netbsd« und einem Neustart ist die Maschine dann als Server einsatzbereit.

Ein NFS-Server braucht selbst kein NFS-Dateissystem im Kernel, also kann folgende Zeile der Kernelkonfiguration ruhig auskommentiert sein, es sei denn, man will den NFS-Server gleichzeitig auch als NFS-Client (dann bei einem dritten NFS-Server) einsetzen):

```
file-system     NFS                 # Network File System client
```

Das Bedienen von NFS-Requests (im wesentlichen das Weiterleiten von NFS-Requests, die via RPC über das Netzwerk empfangen werden, an den Kernel) wird – bis auf das später zu beschreibende Locking – von zwei Dämonen übernommen: *mountd* und *nfsd*. *mountd* prozessiert lediglich Mount-Requests, wohingegen der *nfsd* für das eigentliche Bearbeiten der NFS-Requests zuständig ist.

Der Daemon *mountd* erfüllt im wesentlichen zwei Funktionen: er hört auf *mount*-Anforderungen von NFS-Clients und bedient diese; außerdem importiert er die Erlaubnisspezifikationen aus einer Konfigurationsdatei (»/etc/exports«) in den Kernel. Wichtig ist zu wissen, daß *mountd* den Inhalt von der im folgenden beschriebenen Datei »/etc/exports« in der Regel nur einmal, nämlich beim Start, liest und in den Kernel importiert. Ändert man den Inhalt dieser Datei, während der *mountd* bereits läuft, muß man den Daemon dazu bewegen, »/etc/exports« erneut zu lesen, indem man ihm ein *SIGHUP*-Signal schickt. Das erreicht man mit folgendem Kommando:

```
# kill -HUP `cat /var/run/mountd.pid`
```

In der Datei »/etc/exports« wird angegeben, welche lokalen Dateisysteme vom NFS-Server über das Netz zur Verfügung gestellt (exportiert) werden sollen. Außerdem können hier Optionen angegeben und der Kreis der Clients begrenzt werden.

Jede Zeile in »/etc/exports« (mit Ausnahme der Kommentarzeilen) spezifiziert die Bedingungen, unter denen ein Directory an einen oder mehrere andere Rechner exportiert wird. Im einfachsten Fall wird in einer Zeile nur ein Verzeichnis angegeben, was dann dazu führt, daß es an die ganze Welt exportiert wird. Wegen der fehlenden Sicherheitsmechanismen von NFS wird hiervon strikt abgeraten.

Die Exportbedingungen können nun durch Modifizierer eingeschränkt oder abgeändert werden. So wird beispielsweise mit der Angabe *-network 192.168.48/24* beziehungsweise *-network 192.168.48 -mask 255.255.255.0* der Zugriff auf ein Subnetz beschränkt.

Normalerweise ist nur das am Zeilenanfang angegebene Verzeichnis zum Export freigegeben. Die optionale Angabe von *-alldirs* erweitert die Freigabe auf alle Unterverzeichnisse.

Einen Schreibzugriff auf exportierte Dateisysteme verwehrt man durch die Angabe der Option *-ro* (read only).

Wie in Kapitel 9.1.1 beschrieben, werden NFS-Zugriffe durch *root* aus Sicherheitsgründen standardmäßig auf den unprivilegierten Benutzer *nobody* abgebildet. Durch die Angabe von *-maproot=user* läßt sich diese Abbildung ändern. Insbesondere läßt *-maproot=root* zu, daß *root* auf dem Client auf *root* auf dem Server abgebildet wird. Das ist zwar gefährlich, manchmal (insbesondere beim Bauen und Installieren von Software in einem NFS-Dateissystem) nützlich. *user* kann hier wahlweise mit dem Accountnamen oder über die UID angegeben werden.

Mit Hilfe der Option *-mapall=user* ist es möglich, NFS-Requests aller Benutzer (einschließlich *root*) auf einen Generalaccount abzubilden. Die allgemeinere Form *-mapall=user:group1:group2:...* erlaubt es außerdem, eine oder mehrere Gruppen anzugeben, als deren Mitglied *user* angesehen wird.

Hier nun ein paar Beispiele:

```
/usr -ro -network 131.104.48 -mask 255.255.255.0
```

Exportiert »/usr« (nur Lesezugriffe erlaubt) an das Subnetz 131.104.48.*. Mounten von »/usr/local« ist aber auf den Client wegen des fehlenden *-alldirs* nicht erlaubt.

```
/home -alldirs -maproot=root -network 131.104.48/24
```

Liberaler Export von »/home« an das Subnetz 131.104.48.*.

```
/projekt/src -ro -mapall=nobody
```

Die ganze Welt darf lesend mit den Rechten von *nobody* »/project/src« mounten.

Dies sind die wichtigsten Exportoptionen. Wer weitere eher selten verwendete Optionen benötigt (beispielsweise *netgroups*) sei auf die Manualseite *exports*(5) verwiesen.

Schließlich und endlich kommen wir zu dem Daemon, der im Betrieb die meiste Arbeit zu leisten hat: dem *nfsd*, der für das Abarbeiten aller NFS-Anforderungen zuständig ist. Seine Implementation ist interessant: nachdem *nfsd* die üblichen Arbeiten eines Unix-Daemons erledigt hat (Interpretieren der Kommandozeile und so weiter), führt er einen Systemaufruf (Aufruf von *nfssvc*(2)) durch, aus dem er (mit ganz wenigen Ausnahmen) nie zurückkehrt. Auf diese Weise war es den BSD-Implementatoren auch ohne das Vorhandensein von Kernel-Threads möglich, über den Umweg von Prozessen, die ihren Kontext bereitstellen, ein gut funktionierendes Multithreading zu erreichen. Dabei übernimmt die Routine *nfsrv_rcv()* im Kernel über einen Upcall die Aufgabe, die Last an die *nfsd* zu verteilen.

Üblicherweise werden vier bis sechs Instanzen des *nfsd* gestartet, meistens ist für normale Anwendungen der Standardwert von vier ausreichend. In Kapitel 9.1.6 wird erklärt, wie man erkennen kann, ob eine ausreichende Zahl von *nfsd*-Instanzen gestartet wurde.

Damit alle auf dem NFS-Server benötigten Daemons beim Booten automatisch gestartet werden, sollte man folgendes in die »/etc/rc.conf« eintragen:

```
rpcbind=YES
rpcbind_flags="-l"
nfs_server=YES
nfsd_flags="-6 -t -u -n 4"
```

Die Optionen von *nfsd_flags* bedeuten folgendes:

Höre auf IPv6 Anforderungen (*-6*) (falls IPv6 nicht vorhanden ist, wird einfach IPv4 benutzt); bediene NFS-Clients über TCP (*-t*); bediene NFS-Clients über UDP (*-u*); Starte vier *nfsd*-Instanzen (*-n 4*). Diese *nfsd_flags* sind übrigens die Voreinstellung von NetBSD 1.5.3 und 1.6, man findet sie in kompakterer Form in der »/etc/defaults/rc.conf« wieder als:

```
nfsd_flags="-6tun 4"
```

9.1.3 NFS-Client einrichten

Auch beim Einrichten eines NFS-Clients prüfen wir zuerst, ob der aktuelle Kernel die erforderliche NFS-Unterstützung (diesmal als Client) bereitstellt:

```
% nm -o /netbsd | grep nqnfs_clientd | wc -l
```

Wird eine 1 angezeigt, ist die NFS-Client-Funktionalität im Kernel vorhanden, andernfalls muß sie in der Kernel-Konfigurationsdatei hinzugefügt werden:

```
file-system        NFS                  # Network File System client
```

Nach dem Übersetzen und dem Installieren des neuen Kernels steht dann nach dem ersten Reboot die erforderliche NFS-Client-Funktionalität zur Verfügung.

Es muß nun noch geprüft werden, ob der Mountpunkt (im folgenden Beispiel »/mntpkt«), auf den das entfernte Dateisystem montiert werden soll, vorhanden ist; andernfalls muß der Mountpunkt noch angelegt werden:

```
# ls -ald /mntpkt
ls: /mntpkt: No such file or directory
# mkdir /mntpkt
```

Nun kann das entfernte Dateisystem montiert werden:

```
# mount -t nfs fern1:/home/fritz /mntpkt
```

Ab nun steht der Dateibaum »/home/fritz« des entfernten Rechners »fern1« unterhalb des Mountpunkts »/mntpkt« zur Verfügung.

Beim obigen *mount*-Befehl kann die Option *-t nfs* auch weggelassen werden, weil *mount* so intelligent ist, daß es an der Syntax *host:directory* erkennt, daß ein NFS-Mount gemeint ist. Wenn dieser Dateibaum nicht mehr benötigt wird, kann er mit Hilfe von

```
# umount /mntpkt
```

wieder entfernt werden. Das funktioniert allerdings nur, wenn in dem noch eingebundenen Dateisystem keine Datei mehr offen ist. Es darf also keinen Prozeß mehr geben, der eine Datei unterhalb von »/mntpkt« geöffnet hat. Auch reicht es aus, daß ein Prozeß sein aktuelles Arbeitsverzeichnis unterhalb von »/mntpkt« hat, damit *umount* den Dienst versagt. Wenn Ihnen das passiert, lesen Sie weiter unten unter »Probleme finden und beseitigen« nach, was zu tun ist.

Der *mount*-Befehl ist ein sogenanntes Frontend zu einer Vielzahl spezialisierter Mount-Befehle. Im Fall von NFS-Mounts interpretiert *mount* die Kommandozeile, liest die Datei »/etc/fstab« und ruft letztendlich den Befehl *mount_nfs* auf. Im Prinzip könnte der Benutzer *mount_nfs* auch direkt aufrufen, allerdings überläßt man besser dem generischen *mount* das Interpretieren der Argumente (damit diese über alle Mount-Befehle hinweg einheitlich gehandhabt werden).

Zum NFS-*mount*-Kommando gibt es eine Reihe von Optionen, von denen insbesondere die Attribute *hard*, *soft* und *interruptible* von Interesse sind. Anders als bei lokalen Platten sind Störungen von NFS-gemounteten Dateisystemen leider häufig (wegen Ausfall des NFS-Servers oder einer zwischenliegenden Netzwerkkomponente). Je nachdem, für welches der erwähnten Attribute man sich entschieden hat, reagiert das Betriebssystem unterschiedlich auf NFS-Anforderungen an hängende NFS-Server.

Da gibt es zunächst einmal das »harte« Standardverhalten: Datei-Systemaufrufe blokkieren (und können nicht unterbrochen werden), falls der entfernte NFS-Server hängt. Viele Programme sind auf transiente Fehler nicht vorbereitet und würden unnötigerweise abbrechen, falls eine Dateioperation mit einem transienten Fehler zurückkehren würde. Dies macht sich natürlich insbesondere bei lange laufenden Programmen negativ bemerkbar. Das Standardverhalten sorgt also dafür, daß Programme bei temporären Netzwerkstörungen nicht abbrechen, sondern blockieren, bis die Normalität wieder hergestellt ist.

Das andere Extrem sind die sogenannten »Soft-Mounts«. Sie versuchen, eine bestimmte Anzahl von Wiederholungen der NFS-Anforderungen (standardmäßig 10000) auszuführen, bevor sie dann aber endgültig aufgeben. Der sogenannte Retrycount kann mit Hilfe der Option *-R* von *mount_nfs* eingestellt werden. Wird er zu niedrig gewählt, brechen NFS-Operationen zu früh (zum Beispiel bei überlastetem NFS-Server) ab. Bei zu hohen Retrycount-Werten kann dagegen ein Prozeß zu lange hängen. Mounts werden auf diese »weiche« Weise durchgeführt, indem man die Option *-s* bei *mount_nfs* (beziehungsweise *-o -s* bei *mount*) angibt.

Als von vielen Systemadministratoren bevorzugten Kompromiß bieten sich »interruptible« (abbrechbare) Mounts an. Ihr Verhalten ähnelt denen des »Hard«-Mounts mit dem Unterschied, daß bei unterbrechbaren Mounts geprüft wird, ob dem auf NFS-Operationen wartenden Prozeß ein Interrupt-Signal geschickt wurde, worauf hin dieser die NFS-Operation unterbricht. Unterbrechbare Mounts werden mit Hilfe der Option *-i* bei *mount_nfs* (beziehungsweise *-o -i* bei *mount*) spezifiziert.

Genauso wie lokale Dateisysteme in »/etc/fstab« eingetragen werden, können und sollten über NFS einzubindende Dateisysteme dort eingetragen sein. In der ersten Spalte sind Rechnername und zu mountendes Verzeichnis in der Form »Rechnername:Verzeichnis« anzugeben. In der zweiten Spalte wird der lokale Mount-Punkt eingetragen, in der dritten natürlich *nfs* als Dateisystem. Schließlich nimmt die vierte Spalte die Optionen auf, zum Beispiel *rw,-i* für unterbrechbare Schreib-/Lese-Mounts. Die fünfte und sechste Spalte sind für NFS-Mounts ohne Bedeutung.

Vielleicht ist Ihnen von anderen unix-ähnlichen Betriebssystemen her der Daemon *nfsiod* bekannt. Aufgabe dieses Daemons ist es, auf der Seite des NFS-Clients NFS-Operationen im Hauptspeicher zwischenzuspeichern. Dafür schaltet sich der *nfsiod* zwischen Systemaufrufebene und NFS-Ebene. So gehen beispielsweise alle *read*(2)-Aufrufe erst an den *nfsiod*, bevor sie über das Netz an den NFS-Server weitergeleitet werden. Der *nfsiod* prüft in einem solchen Fall, ob er die Leseanfrage aus seinem eigenen Cache befriedigen kann. Falls nein, formuliert *nfsiod* die Anfrage gegebenenfalls so um, daß sie eine hinreichend große Datenmenge umfaßt. Auf diese Weise wird das leistungssteigernde »Read ahead caching« erreicht. Die Leistungssteigerung ist dabei darauf zurückzuführen, daß für mehrere kurze Leseanforderungen nur ein (aufwendiger) RPC-Request ausgeführt werden muß, alle nachfolgenden Leseanforderungen können danach aus dem Cache bedient werden.

Genau wie bei den *nfsd*-Daemons werden mehrere *nfsiod* gestartet (üblicherweise vier bis sechs), um quasi-gleichzeitige Leseanforderungen (von verschiedenen Prozessen) besser bedienen zu können.

Seit Betriebssystemversion 1.5 gibt es in NetBSD allerdings keinen *nfsiod* mehr. Dessen Funktionalität wurde, nachdem Kernel-Threads eingeführt worden waren, vollständig in den NFS-Teil des Kernels übernommen. Angaben wie

```
nfsiod=YES
```

oder

```
nfsiod_flags=
```

sind in »/etc/rc.conf« nicht mehr erforderlich oder wünschenswert und sollten daher entfernt werden.

9.1.4 Selten benötigt

mount_umap

Möchte man Dateisysteme aus unterschiedlichen administrativen Domänen via NFS miteinander verbinden, steht man vor dem Problem, daß Benutzer, die Accounts in beiden Domänen besitzen, in der Regel unterschiedliche UIDs haben. Sie können also normalerweise nicht als Eigentümer auf ihre via NFS gemounteten Dateien zugreifen. Zur Lösung dieses Problems stellt NetBSD das Kommando *mount_umap* zur Verfügung.

Der Administrator gibt (über die Optionen *-u* beziehungsweise *-g*) *mount_umap* zwei Dateien an, in denen die Abbildungen der UIDs und der GIDS von der einen auf die andere Domäne enthalten sind. *mount_umap* montiert dann das angegebene Zielsystem unterhalb eines ebenfalls angegebenen Mountpunkts. Wichtig: Damit der Befehl *mount_umap* funktioniert, muß der aktuelle Kernel mit der Option

```
file-system    UMAPFS          # NULLFS + uid and gid remapping
```

übersetzt worden sein!

Die Benutzung von *mount_umap* sei hier anhand eines Beispiels erläutert. Nehmen wir an, auf dem lokalen NetBSD-System gäbe es einen Account *fritz* mit der UID 777 und der GID 15. Fritz hat auch einen Account *xy123* (mit der UID 123, GID 123) auf der Maschine *most.secret.com*, von der wir das Dateisystem »/home« lokal unter »/mnt« einbinden wollen. Zunächst legen wir als *root* die Dateien »uidfile«

```
1
777 123
```

und »gidfile«

```
1
15 123
```

an. Die erste Zeile gibt die Zahl der Einträge an, gefolgt von paarweisen Einträgen der Form *<lokale ID> <Original-ID>*. »uidfile« und »gidfile« müssen *root* gehören und dürfen nur von *root* beschreibbar sein.

Zunächst wird »/home« normal via NFS eingebunden (dazu muß dieses Verzeichnis natürlich auf *most.secret.com* exportiert werden):

```
# mount most.secret.com:/home /mnt2
```

Dann wird »/mnt2« erneut gemountet, diesmal mit *mount_umap*:

```
# mount_umap -u uidfile -g gidfile /mnt2 /mnt
```

Nun kann *fritz* unter »/mnt« auf seine Dateien als Eigentümer zugreifen.

lockd

NFS bietet standardmäßig die meisten Dateioperationen (*open*, *close*, *read*, *write*, *stat* und so weiter) für ein entferntes Dateisystem an, aber eben nicht alle. Stiefkind der Implementation ist das File Locking.

NetBSD bietet auf Systemaufrufebene zwei Methoden an, eine Datei gegen Zugriffe von anderen Prozessen zu sperren: die Systemaufrufe *flock*(2) und *fcntl*(2). Dabei bietet *flock*(2) sogenannte »Advisory Locks« an, das heißt, Prozesse müssen kooperieren, um einen geregelten Dateizugriff zu gewährleisten (beide müssen *flock*(2) aufrufen). Im Gegenteil dazu steht *fcntl*(2), mit dem man eine ganze Datei oder Teile von ihr gegen den Zugriff anderer Prozesse sperren kann. Die mit *fcntl*(2) erhaltenen Locks sind absolut, das heißt, jeder andere Prozeß, der versucht, auf die gesperrte Datei zuzugreifen, blockiert solange, bis der Lock wieder freigegeben wird. Es gibt noch einen weiteren wichtigen Unterschied zwischen den mit *flock*(2) und *fcntl*(2) erwirkten Locks, der deren Freigabe betrifft: Es kann sein, daß ein Lock auf eine Datei von mehreren Prozessen angefordert wird. Bei *flock*(2) wird der Lock erst dann wieder aufgehoben, wenn der letzte Lock wieder freigegeben wird. Anders bei *fcntl*(2): Hier reicht es aus, daß irgendein Prozeß den Lock freigibt, zum Beispiel dadurch, daß er den Dateideskriptor mit *close*(2) schließt, um den Lock aufzuheben. Dieses unsinnige Verhalten von *fcntl*(2) wird leider vom POSIX-Standard erzwungen. Beide Locking-Methoden (*flock*(2) und *fcntl*(2)) können aber völlig unabhängig voneinander eingesetzt werden und beeinflussen sich nicht gegenseitig.

Damit das im NFS-Protokoll nicht implementierte Locking dennoch eingesetzt werden kann, wurde diese Aufgabe einem separaten Daemon, dem *rpc.lockd* (oft auch nur als *lockd* bekannt) übertragen. NetBSD stellt ihn zur Verfügung. Die Sache hat nur einen

kleinen Haken: NetBSD stellt mit dem *rpc.lockd* nur die Implementation der Serverseite zur Verfügung, nicht jedoch die Client-Seite. Das heißt, es gibt unter NetBSD zur Zeit keine Möglichkeit, einen Lock über NFS zu erwirken. Ein NetBSD NFS-Server kann aber NFS-Locking-Anforderungen, die von Maschinen eines anderen Betriebssystem (zum Beispiel Solaris) aus gestellt werden, honorieren.

Wenn man den *rpc.lockd* einsetzt, sollte man ebenfalls den *rpc.statd* benutzen. Dieser Dämon stellt einen Status-Monitoring-Dienst im Verbund mit anderen *rpc.statd* auf anderen Maschinen zur Verfügung. Stürzt eine von *rpc.statd* überwachte Maschine ab und bootet anschließend, meldet sich deren *rpc.statd* beim lokalen *rpc.statd* und dieser kann dann die Programme benachrichtigen, die den Monitoring-Dienst vom *rpc.statd* angefordert haben. Da *rpc.lockd* von diesem Dienst Gebrauch macht, werden also *rpc.lockd* und *rpc.statd* in der Regel immer gemeinsam betrieben.

9.1.5 Probleme finden und beseitigen

Irgend etwas funktioniert mit NFS nicht. Dann gilt es zuerst herauszufinden, bei welcher der drei wesentlichen Komponenten NFS-Server, NFS-Client oder dem Netzwerk dazwischen das Problem liegt. Zur Diagnose stellt NetBSD (wie die meisten anderen Unix-Varianten auch) eine Reihe von Programmen zur Verfügung:

rpcinfo zeigt die im RPC-System registrierten RPC-Dienste an. In der Form

```
# rpcinfo <hostname>
```

werden die auf *hostname* laufenden RPC-Dienste ermittelt. Schlägt zum Beispiel das NFS-Mounten eines Verzeichnisses auf dem entfernten Rechner *flippy* fehl, kann man mittels

```
# rpcinfo flippy
```

überprüfen, ob die benötigten Dienste *portmapper*, *mountd* und *nfsd* auf *flippy* registriert sind.

Als nächstes ist es denkbar, daß alle NFS-Dienste laufen, aber das einzubindende Verzeichnis nicht exportiert ist. Hier hilft der Befehl *showmount* weiter.

```
# showmount -e flippy
```

zeigt an, welche Verzeichnisse *flippy* (an wen) exportiert. Beispiel:

```
# mount flippy:/usr /mnt
mount_nfs: can't access /usr: Permission denied
# showmount -e flippy
Exports list on flippy:
/home                                137.85.061.0
```

Aha, »/usr« ist gar nicht exportiert!

Wichtig ist, Exportlisten grundsätzlich mit *showmount* (und nicht durch visuelles Inspizieren von »/etc/exports«) zu überprüfen. Beispielsweise kann man nach einem Eintrag in »/etc/exports« vergessen haben, dem *mountd* dies mitzuteilen, oder die exports-Datei enthält einen Tippfehler. Benutzt man *showmount,* geht die Information den gleichen Weg wie beim Mounten (über RPC) und diese offensichtlichen Fehler werden sofort gefunden.

Auch das Unmounten kann fehlschlagen, was sogar ein häufiges Problem ist. Nehmen wir beispielsweise an, »flippy:/home« sei auf dem lokalen System als »/mnt« eingebunden und soll nun ausgebunden werden:

```
# umount /mnt
umount: /mnt: Device busy
```

Diese Fehlermeldung besagt, daß es auf dem lokalen System noch mindestens einen Prozeß gibt, der (eine) Datei(en) geöffnet hat. Damit der Prozeß seine Modifikationen zurückschreiben kann, erlaubt es das Betriebssystem dann nicht, das fremde Dateisystem zu entfernen. Wie kann man nun feststellen, welcher Prozeß den Unmount-Vorgang blockiert?

Im Paketsystem »pkgsrc« ist ein populäres Kommando, *lsof* (LiSt Open Files) enthalten (unter »pkgsrc/sysutils/lsof«), das auch von Administratoren unter Linux oder Solaris gerne in solchen oder ähnlichen Fällen benutzt wird. *lsof* kann natürlich hierzu benutzt werden, jedoch bietet die BSD-Unix-Familie bereits direkt als Bestandteil des Betriebssystems ein geeignetes Werkzeug an: *fstat*.

In diesem typischen Fall würde man als NetBSD-Administrator *fstat* folgendermaßen verwenden:

```
# fstat -f /mnt
USER     CMD          PID   FD MOUNT      INUM MODE         SZ|DV R/W
fritz    vi           420   wd /mnt       1984 drwxr-xr-x    1536 r
fritz    vi           420    3 /mnt       1999 -rw-r--r--     292 r
```

Damit ist der schuldige Prozeß identifiziert. Fritz hat noch einen *vi*-Prozeß laufen, der sowohl sein »Working Directory« unterhalb von »/mnt« besitzt, als auch eine Datei geöffnet hat (Deskriptor 3). Nun kann der Administrator Fritz bitten, *vi* zu beenden oder im Notfall den *vi*-Prozeß killen.

9.1.6 Leistungssteigerung von NFS

Meistens ist die Leistung von NFS mit den Standardeinstellungen adäquat. Jedoch kann beispielsweise beim Vorliegen bestimmter Netzwerk- oder Hardwaregegebenheiten mangelnde Leistung zum Problem werden. NetBSD bietet eine Vielzahl von Stellschräubchen, mit denen man das Feintuning von NFS vornehmen kann, um die Leistung zu steigern. Allerdings ist das Tuning eine außerordentlich komplexe Sache, so

daß im Rahmen dieses Buchs nur die groben Züge des Tunings besprochen werden können. Für Details sei auf Fachliteratur, zum Beispiel »Optimizing NFS Performance: Tuning and Troubleshooting NFS on HP-UX Systems« von David Olker (ISBN: 0130428167) verwiesen.

Die wichtigsten Tuningparameter sind:

- TCP oder UDP als Protokoll.
- Zahl der *nfsd* auf dem NFS-Server.
- Größe der Puffer.
- Synchrones/asynchrones Schreiben.

Vor der Therapie steht die Diagnose und das wichtigste Diagnosewerkzeug heißt *nfsstat*. Bei jeder NFS-Operation wird im Kernel ein entsprechender Zähler inkrementiert (sowohl auf dem Client als auch auf dem Server). *nfsstat* dient dazu, diese Zähler auszulesen. Es ist allerdings nicht ganz einfach, aus den von *nfsstat* gelieferten Zahlen eindeutige Empfehlungen für Tuning-Maßnahmen abzuleiten. Bei den folgenden (wichtigsten) Tuning-Maßnahmen kommen wir ohne *nfsstat* aus:

TCP oder UDP als Protokoll

Da UDP den weitaus geringeren Protokoll-Overhead besitzt, ist die NFS-Performance unter optimalen Bedingungen (keine Paketverluste, geringe Netzwerk-Latenz) mit UDP am besten. Sobald jedoch Paketverluste auftreten, kehrt sich dies ins Gegenteil um. Bei der Verwendung von UDP muß die Applikation die korrekte Übertragung der Netzwerkpakete sicherstellen. Dazu startet sie beim Versenden eines RPC-Requests einen Timer. Wenn dann ein Timeout auftritt, wird der gleiche RPC-Request erneut versendet (RPC Retransmit). Standardmäßig werden acht Kbyte große Puffer verwendet, was mit dem für den Headern benötigten Platz dazu führt, daß die UDP-Pakete auf sechs IP-Pakete heruntergebrochen werden müssen. Bei einem Retransmit müssen also wieder sechs neue IP-Pakete verschickt werden. Anders ist es bei der Verwendung von TCP, hier wird die Kommunikationssicherheit vom Protokoll garantiert (und bei Paketverlusten braucht in der Regel nur ein einziges IP-Paket erneut angefordert zu werden).

Aus diesen Gründen ist NFS in der Regel mit TCP ebenso schnell wie mit UDP, es sei denn, die Host-CPU ist besonders langsam, so daß der wesentlich höhere Protokoll-Aufwand von der langsamen CPU nicht verkraftet wird. Dies dürfte bei modernen Rechnern (zum Beispiel bei heute typischen PCs) keine Rolle spielen, ist aber bei einigen von NetBSD unterstützten Architekturen (zum Beispiel der VAX) oder im Embedded-Bereich zu beachten. Aus Gründen der Übertragungssicherheit kommt der Einsatz von UDP sowieso nur in LANs in Betracht, über unzuverlässige Leitungen und in WANs ist die Verwendung von TCP unabdingbar. Leider gibt es dabei ein Problem insoweit als NFS über TCP von den meisten Nicht-BSD-Betriebssystemen nicht unterstützt wird.

Zahl der nfsd

Je größer die Last auf einen NFS-Server ist, desto mehr *nfsd* sollten laufen. Wie in Kapitel 9.1.2 erläutert, sind die *nfsd* quasi Userland-Frontends zum Systemaufruf *nfssvc*(2), und *nfsrv_rcv* im Kernel kann nun die Last auf die *nfsd* aufteilen. Dabei wird die Last nicht gleichmäßig auf alle *nfsds* verteilt, sondern die *nfsds* sind in einer Queue arrangiert und ein Request wird an den ersten nicht beschäftigten *nfsd* durchgereicht. Ein bestimmter *nfsd* übernimmt also die Hauptlast, die anderen werden nur aktiv, wenn die in der Queue vorher kommenden beschäftigt sind.

Wegen dieses Schedulingverfahrens ist es ein Leichtes festzustellen, ob eine ausreichende Zahl von *nfsd* laufen. Zum Beispiel kann man mit Hilfe von *top* kontrollieren, wieviele *nfs*-Daemons gerade CPU-Zeit bekommen. Sind alle Daemons gleich stark beschäftigt, sollte die Zahl der *nfsds* erhöht werden. Genauer wird diese Methode, wenn man nach einer gewissen Laufzeit die erhaltene CPU-Zeit der *nfs*-Daemons auf einem NFS-Server mit *ps* ermittelt:

```
# ps auxww | grep nfsd
root   155  0.0  0.1    68    76 ?? Is   23Aug02   0:00.05 nfsd: master
root   157  0.9  0.1    48   160 ?? SL   23Aug02   1:29.73 nfsd: server
root   159  0.0  0.1    48   160 ?? SL   23Aug02   0:03.99 nfsd: server
root   160  0.0  0.1    48   160 ?? SL   23Aug02   0:01.19 nfsd: server
root   158  0.0  0.1    48   160 ?? SL   23Aug02   0:00.21 nfsd: server
```

In der obigen Liste sind die nfs-Daemons bereits nach erhaltener CPU-Zeit sortiert. Man erkennt, daß der *nfsd* mit der PID 157 bei weitem die meiste CPU-Zeit erhalten hat. Das entscheidende ist aber, daß der letzte Eintrag in der Liste, der *nfsd* mit der PID 158, so gut wie nie gebraucht wurde. Die Zahl der *nfsds* ist also ausreichend.

Größe der Puffer

Standardmäßig benutzt NFS über UDP acht Kbyte große Puffer, bei NFS über TCP sind diese 64 Kbyte groß. Falls es die Netzwerkhardware unterstützt, kann das Vergrößern der Puffergröße vor allem bei UDP bei einen Leistungsgewinn ergeben, da man im Mittel mit weniger RPCs auskommt.

Beim Vorhandensein alter Hardware hilft das Gegenteil: Durch die Verringerung der Größe der Puffer erreicht man, daß die NFS-Puffer vollständig in die Hardwarepuffer auf der Netzwerkkarte passen. Beispielsweise bei alten Western Digital/SMC Elite Ultra-Karten kann es sinnvoll sein, die NFS-Puffergröße auf vier Kyte zu halbieren (Details erfährt man mit *man we*).

Die Puffergröße kann mit Hilfe der Optionen *-r* und *-w* für *mount_nfs* beziehungsweise in der »/etc/fstab« im Options-Feld angegeben werden. Ein Beispiel aus der »/etc/fstab«:

```
flippy:/home    /mnt     nfs    rw,noauto,-r=4096,-w=4096    0    0
```

Noch sinnvoller ist es in einem solchen Fall, den Standardwert für die Größe der NFS-Puffer direkt im Kernel festzulegen, da diese Einstellung ja für jeden NFS-Mount gewählt werden soll. Dies erreicht man, indem man (mit den Werten des obigen Beispiels) folgende Zeilen in seine Kernel-Konfigurationsdatei einträgt und einen neuen Kernel baut:

```
options NFS_RSIZE=4096
options NFS_WSIZE=4096
```

Synchrones oder asynchrones Schreiben

Zum synchronen und asynchronen Schreiben gibt es nicht viel zu sagen, außer daß man die Schreibgeschwindigkeit für alle Mount-Arten durch asynchrones Schreiben auf Kosten der Datensicherheit steigern kann. Beim asynchronen Schreiben kehrt der Systemaufruf *write*(2) zurück, sobald er den Schreibauftrag abgeliefert hat, aber noch vor der Beendigung der Schreibvorgangs. Gezwungenermaßen muß er den Schreiberfolg zurückmelden und die Applikation fährt in dem festen Glauben fort, die Daten seien sicher gespeichert. Stürzt aber beispielsweise der NFS-Server just in diesem Moment ab, sind Daten nicht auf Platte gespeichert.

Außer in ganz bestimmten Ausnahmefällen, in denen man sich solche fehlende Datenintegrität leisten kann, sollte man daher auf die Verwendung asynchronen Schreibens verzichten.

9.2 Der Samba-Server

von Thorsten Lindloff

Eine Musik, die in die Beine geht und ein Gedanke an den Karneval in Rio de Janeiro mag die Entwickler dieses Pakets dazu bewogen haben, ihrem Programm diesen Namen zu geben und bei den Benutzern entsprechende Gedanken zu wecken. »Maccarena« allerdings würde aus der Sicht mancher Mitmenschen im Hinblick auf das zuweilen recht eigensinnige Verhalten einiger Windows-NT-Installationen manchmal besser passen. Der Begriff »Samba« ist in Wirklichkeit eine Verballhornung von »SMB«, Server Message Block, dem eigentlichen Namen dieses Protokolls.

Diese einleitenden Worte beschreiben schon, wer Samba braucht. Dieser Server dient dem Datenaustausch von Unix-Maschinen mit Windows-PCs. Immerhin laufen zirka neunzig Prozent aller Desktop-PCs unter einer der Windows-Versionen. Also ist es äußerst wichtig, daß es eine Möglichkeit zum Datenaustausch mit ihnen gibt.

Normalerweise werden Verzeichnisse unter Unix über NFS eingebunden und es gibt auch NFS-Clients und NFS-Server für Windows, sowohl freie als auch kommerzielle. Auch das FTP-Protokoll läßt sich über eine Vielzahl von Client-Programmen windowsseitig einsetzen, in die »normale« Windows-Landschaft passen diese Lösungsansätze aber nicht übermäßig gut. Der durchschnittliche Windows-Anwender wird immer die Lösung präferieren, die sich nahtlos in das System einpaßt.

Um mit Windows-Rechnern auf die Verzeichnisdienste im Unix-Netzwerk zugreifen zu können, entstand 1992 der Samba-Server, der einen NT- oder 2000-Server in etlichen Funktionen ersetzen kann. So kann man mit Hilfe von Samba auch einen Windows-9x-/Millenium-Rechner als Druckserver für Unix-Systeme einrichten. Das geht allerdings nicht ohne einigen Aufwand hinsichtlich Zeit und Geld für die Investitionen auf der Windows-Seite. Viel interessanter als der Druckdienst auf einer derartigen Installation ist allerdings das, was Samba sonst noch zu bieten hat.

Das SMB-Protokoll ist eigentlich ein Überbleibsel aus einer Produktlinie, die von IBM und einer Firma namens Systec entwickelt wurde, um für die damals neuen PCs eine Netzwerkhardware anbieten zu können. Genannt wurde diese Produktlinie, die aus den Netzwerkkarten, den dazugehörigen Kabeln und einem Treiber namens NetBIOS bestand, »Broadband LAN Product«.

Diese Produkt hatte allerdings auch ein paar Nachteile, die sich wie ein roter Faden durch das SMB-Protokoll und die Windows-NT-Netzwerk-Philosophie ziehen. Beispielsweise war die Größe des Netzwerks auf 80 Rechner begrenzt und die Datenpakete waren nicht routingfähig.

Anstatt das NetBIOS-Protokoll sterben und in Frieden ruhen zu lassen, verwendeten etliche Anbieter von Netzwerksoftware die NetBIOS-Schnittstelle, um eigene Protokolle darauf aufzusetzen. Dazu gehören DECNet, IPX/SPX und SNA. NetBIOS über TCP/IP wird in manchen Kreisen auch NBT genannt. Wie NBT funktioniert, wird in den RFCs 1001 und 1002 beschrieben, die etlichen Informatikern wahrscheinlich auch als Internet-Standard Nummer 19 bekannt sind.

Das SMB-Protokoll wurde für die NetBIOS-Schnittstelle als ein PC-Netzwerkprotokoll entwickelt, um Pakete zu versenden und zu empfangen.

Das hat sich bis zum Erscheinen von Windows 2000 auch nicht geändert. Windows 2000 ist übrigens das erste Microsoft-Betriebssystem, das seine Pakete über TCP/IP ohne die NetBIOS-Kapselung verschickt.

Bei NetBIOS handelt sich im wesentlichen um ein Netzwerkprotokoll, das von Microsoft für deren Betriebssysteme weiterentwickelt wurde; wahrscheinlich aus dem Grund, die Kundschaft zu nötigen, möglichst die gesamte EDV auf diese Systeme umzustellen. Im Unterschied zu den anderen Protokollen gibt es davon nämlich keine offenen Quellcodes, zumindest nicht von Microsoft. Die Ausnahme stellt Samba dar, das allerdings aus Kompatibilitätsgründen in der Aktualität immer etwas hinter den Microsoft-Systemen herhinkt. Das ist allerdings kein Qualitätsmangel von Samba, sondern ein Microsoft-Problem. Und die Sicherheitsprobleme von Microsoft-Betriebssystemen sind ja ohnehin schon fast legendär.

NetBIOS setzt auf die Transportschicht auf und kann über einige andere Transportprotokolle übertragen werden. Es handelt sich dabei um TCP/IP, IPX/SPX für Novell-Netzwerke und das von Microsoft entwickelte NetBEUI. NetBIOS-Daten lassen sich auch in TCP/IP-Pakete kapseln. Damit kann man alle Zugriffe auf das Netzwerk, unabhängig davon, ob sie intern oder extern sind, über dasselbe Transportprotokoll übertragen, was nicht nur die Administration eines heterogenen Netzwerks sehr vereinfacht, sondern auch der Stabilität und Performance des Gesamtsystems dient.

Auf NetBIOS baut das SMB-Protokoll auf. Mit SMB lassen sich in einem Netzwerk auf verschiedenen Rechnern diverse Verzeichnisse freigeben und auf anderen Maschinen unter Angabe des Rechner- und des Freigabenamens sowie eines Passworts in die Verzeichnisbäume anderer Maschinen einhängen oder als virtuelle Laufwerke importieren. Außerdem lassen sich über das SMB-Protokoll die Drucker im Netzwerk von mehreren Windows-Rechnern aus nutzen, wobei es sich hier primär um Maschinen handelt, die auf einer 9x-/ME-Version laufen.

Ob eine Konfiguration an der Konsole einfacher ist als wenn man diese Arbeit mit der Maus erledigt, muß jeder Anwender letztlich für sich selbst beurteilen. Mit Samba können Sie die Konfiguration sowohl im Webbrowser mit SWAT, an der Konsole mit Hilfe eines Editors und mit *ksamba*, einem Administrationstool für die KDE-Benutzeroberfläche oder *gtksamba* durchführen. Beide Programme sind aber nicht besonders gut. Eine Alternative stellt aber auch die webgestützte Verwaltung mit Webmin (Kapitel 15) dar. Im Webmin ist SWAT integriert.

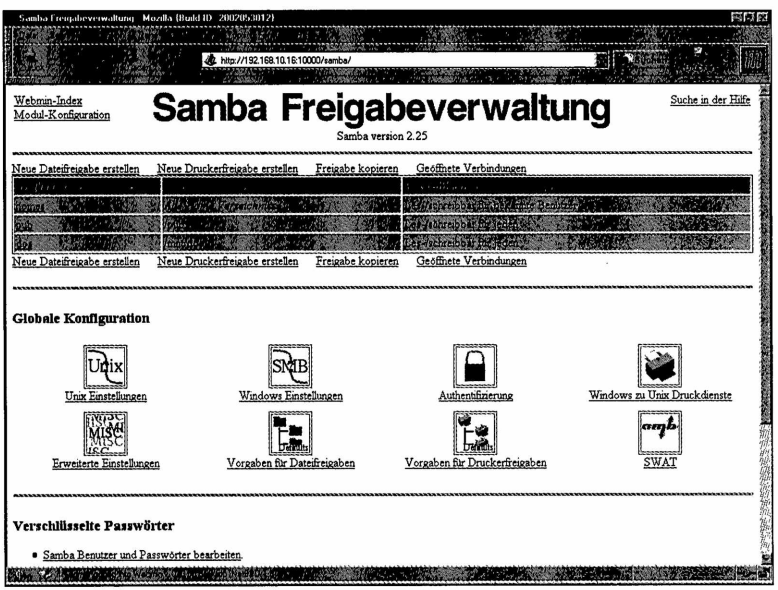


Abb. 9.1: Samba-Verwaltung in Webmin von einer Windows-Station aus

Wenn Sie als Privatperson oder zu Testzwecken einen solchen Windows-Server aufsetzen wollen, wird Ihnen schon aus Kostengründen kaum eine andere Wahl als Samba bleiben.

Die Vorteile eines solchen Systems liegen allerdings, zuerst einmal ungeachtet der Frage, ob es nun Samba oder ein echter NT-Server werden soll, auf der Hand. Wenn Sie mit reichlich Hardware gesegnet sind und sogar ein Raidframe unter NetBSD betrei-

ben, haben Sie so den Vorteil, die Daten zentral auf einem Fileserver ablegen zu können und auf einer anderen Maschine zu arbeiten. Die Datensicherung können Sie dann ebenfalls zum Beispiel mit Hilfe eines Shellskripts zentral vornehmen. Das Schema dafür ist: Auf der langsameren(!), älteren Maschine läuft der Server, der die Daten zur Verfügung stellt, und auf der schnelleren, auf der die Anwendungen installiert sind, bearbeiten Sie diese Daten. Wenn Sie nur Daten von einer Festplatte in ein Netzwerk schaufeln wollen, reicht in der Transferleistung ein älterer Rechner mit einem 200-MHz-Prozessor und einer größeren UDMA-33-Festplatte mit mehr als acht Gigabyte Speicherplatz bereits aus, um ein kleineres 100-MBit-Netz mit Daten zu versorgen, wenn der Server nichts anderes tut, als seine Daten zur Verfügung zu stellen.

Ein weiterer Vorteil von Samba sollte nicht unerwähnt bleiben: Wenn Sie unter Microsoft-Betriebssystemen Konfigurationsänderungen vornehmen, müssen Sie den Rechner danach neu starten. Wenn Sie ein Unix-System wie NetBSD besitzen, müssen Sie das nicht. Das Tempo vom Anhalten und Neustarten des Samba-Servers hängt in erster Linie von Ihren Tippkünsten ab. Sie müssen den Benutzern in Ihrem Netz nur bekanntgeben, wann Sie Samba neu starten.

Abzuwägen sind allerdings auch andere Argumente: Sie sollten wissen, daß ein Samba-Server erfahrungsgemäß etwa um die Hälfte langsamer als das Windows-NT-Gegenstück ist. Außerdem sollen Sie sich zumindest mit Ihrem Windows-System auskennen, die Grundlagen für den Rest lesen Sie hier.

Die essentielle Voraussetzung für den Betrieb von Samba ist, daß alle Windows-Clients als Transportprotokoll TCP/IP und nicht das proprietäre NetBEUI verwenden. Mit den 32-Bit-Versionen von Windows wird TCP/IP ohne weiteres funktionieren. Für die alten Windows-3x-Betriebssysteme gibt es auf der Microsoft-Website einen speziellen Treiber. Sie müssen nur überprüfen, ob er installiert ist und ihn gegebenenfalls nachinstallieren. Wenn Sie wollen, daß sich die Benutzer in Zukunft auf einem Samba-Server einloggen, um ihre Daten darauf zu verwalten, sollten Sie den NetBSD-Rechner als Domain Controller einrichten.

9.2.1 Das Domänensystem von Windows

Der Begriff »Domain« ist eigentlich ein aus der Unix-Welt entliehener Begriff, der bei Microsoft etwas ganz anderes bedeutet: Microsoft definiert eine NT-Domain als eine Ansammlung von einzelnen (Windows-NT-) Rechnern, die untereinander über verschiedene Rechte in dieser Domain verfügen. Das wesentliche Recht dabei ist, Mitglied dieser Domain zu sein. Eine NT-Domain ist zuerst einmal als eine Struktur anzusehen, in der zentrale Administrationsaufgaben an ausgewählte Mitglieder der Domain vergeben werden. Diese Aufgaben sind, wie auch unter Unix-Systemen, sehr vielfältig. Es geht dabei vor allem um die Verwaltung der Benutzer und der sogenannten Browser-Dienste.

Es gibt noch einen weiteren Unterschied zu Unix-Netzen, der nicht unerwähnt bleiben sollte: diese Netze sind im mathematischen Sinne linear, Unix-Systeme sind hierarchisch. Das Ende der Microsoft-Welt liegt deshalb theoretisch gleich an der Grenze der Domain. Eine Ausnahme stellen die »Trusted Domains« dar, die eine gewisse Erweiterung des Microsoft-Netzes ermöglichen.

Die Subdomain

Auch bei den flachen Netzwerkstrukturen von Microsoft gibt es den Begriff der »Subdomain«. Eine NT-Domain wird in logisch unabhängige Subdomains unterteilt, in der es jeweils einen sogenannten »Local Master Browser« gibt, der das Browsing erledigt. Wenn die Daten über die Subdomain hinaus weiterverteilt werden sollen, braucht man zusätzlich noch einen »Domain Master Browser«. Nur der Namensraum und damit auch die Topologie bleiben flach: In den verschiedenen Subdomains dürfen sich keine identischen User- oder Hostnamen befinden, auch wenn sie nicht in der jeweiligen Subdomain liegen.

Die Workgroup

Eine Arbeitsgruppe ist wieder eine Erfindung von Microsoft und bezeichnet eine Sammlung von Rechnern, die in einem Netz miteinander zu tun haben und eigentlich nur Daten untereinander austauschen und sich einen oder auch mehrere Drucker teilen. In einem solchen Netz, das von Microsofts Denkansatz aus betrachtet nur mit Windows-9x-Rechnern funktionieren soll, gibt es keine Automatismen wie eine zentrale Benutzerdatenbank in der Domain. Innerhalb der Gruppe benutzen alle Rechner den gleichen Arbeitsgruppennamen. Außerdem ist jeder einzelne Rechner im Netz mit einem Benutzer gleichbedeutend. Das bedeutet, daß, auch wenn es einen »Benutzermanager« unter Windows 9x/ME gibt, diese Rechner schon im Ansatz keine Multiuser-Systeme sind. Das können Sie live und direkt verfolgen, wenn Sie einmal mehrere Konten auf so einer Maschine einrichten und dann die Daten eines beliebigen Mitbenutzers testweise in den Papierkorb werfen und löschen. Bedingung für diese Spielerei ist aber, daß die Maschinen in der Windows-Standardkonfiguration laufen (jeder hat Vollzugriff auf die Daten). Machen Sie das aber bitte nicht mal testweise bei Ihrem Arbeitgeber oder gar im Internet, sondern zu Hause! In einem solchen Netz sind eben alle Rechner und alle Benutzer in jeder Hinsicht gleichberechtigt.

Die verschiedenen Rollen der Server in einem Windows-Netzwerk

Wenn Sie schon einmal einen NT-Server aufgesetzt haben, wird Ihnen aufgefallen sein, daß die Bezeichnung »Server« auf der Softwareverpackung eigentlich ein ungenauer Begriff für einen NT/2000-Server ist. Ein derartiger Server kann in einem Netzwerk verschiedene Rollen spielen:

Es gibt einen gewöhnlichen NT-Server, der in erster Linie als File- oder Printserver dient. Dann gibt es noch einen Primary Domain Controller (PDC) und einen Backup Domain Controller (BDC). Jeder dieser Server hat eine andere Aufgabe.

Ein gewöhnlicher File- oder Printserver besitzt in einem NT-Netz normalerweise keine Funktionalitäten zur Verwaltung einer Benutzerdatenbank. Das ist immer die Aufgabe des PDC. Das bedeutet, daß in einem echten NT-Netz immer eine Maschine vorhanden sein muß, die diese Aufgabe übernimmt. In kleineren Netzen kann der ein solcher Server auch die anderen Dienste mit übernehmen.

Die Aufgabe des BDC ist es, eine Kopie der Datenbank des PDC vorzuhalten, die im laufenden Betrieb mit der Datenbank im PDC abgeglichen wird. Er übernimmt die

Arbeit des PDC, wenn dieser ausfällt, automatisch und ohne Zutun des Administrators. Ein BDC muß allerdings nicht zwingend vorhanden sein.

Hinzu kommen noch die sogenannten »Browser«, die sämtliche Daten über vorhandene Dienste im Netzwerk einsammeln und veröffentlichen. Dabei geht es vor allen Dingen um die Druck- und Filesharing-Dienste.

9.2.2 Einschränkungen von Samba

Wenn Sie einen Samba-Server in eine vorhandene NT-Domain integrieren wollen, können Sie Ihren Samba-Server nicht gleichzeitig als Domain Master Browser konfigurieren. Standardmäßig ist ein NT-PDC (Primary Domain Controller) für eine Domain gleichzeitig der Domain Master Browser für diese Domain.

Für Subnetze, die nicht zu diesem NT-PDC gehören, können Sie Samba als Local Master Browser aufsetzen, ohne daß es Probleme gibt.

Eine weitere Einschränkung ist, daß Sie einen Samba-Server nicht als echten Backup Domain Controller einsetzen können. Auch das ist kein wirklicher Makel, sondern liegt darin, daß Microsoft die dazu notwendigen Protokolle nicht freigibt, mit denen die Daten auf den Maschinen synchronisiert werden. Damit weiß eben auch niemand, wie das Ganze genau funktioniert.

Aus dem gleichen Grund ist Samba auch nicht in der Lage, einen Exchange-Server zu ersetzen. Auch dazu gibt es Alternativen.

Sie können allerdings die Policies für NT/2000/XP-Clients damit verwalten. Domain-Logins für alle Windows- Systeme ab 95 sind ebenfalls möglich. Außerdem können Sie die lokalen Profildateien für die verschiedenen Benutzer auf Ihrem Samba-Server lagern und damit die Administration Ihres Windows-Netzes vereinfachen.

9.2.3 Die Namensauflösung in Windows-Netzen

Sie können das Ganze, ähnlich wie bei Unix-Systemen, mit einer »hosts«-Datei, die dort »lmhosts« heißt, erledigen, was allerdings bereits bei wenig mehr als zwei Rechnern ziemlich kopflastig wird. Auf jedem Rechner im Netz muß nämlich eine solche Datei gepflegt werden, die außerdem auf allen Systemen im Windows-Netz identisch sein muß.

Ein anderer Weg ist das WINS-System (Windows Internet Name Server), das ähnlich funktioniert wie das »richtige« DNS, allerdings nicht dazu kompatibel ist.

Wenn Sie Ihren Windows-Rechner so konfiguriert haben, daß er sich bei einem WINS-Server anmeldet, wird er im Laufe der Zeit eine komplette Tabelle mit den vorhandenen NetBIOS-Rechnernamen in der Arbeitsgruppe und deren IP-Adressen erhalten.

Eine Änderung der Namensauflösung hat in Windows-2000-Systemen stattgefunden. Diese Systeme benutzen DNS und bieten WINS nur noch aus Kompatibilitätsgründen zu den alten NT- und 9x-Systemen.

Ohne WINS

Wenn ein Client eine Verbindung zu einem Server sucht, fragt er per Broadcast zuerst einmal im Netzwerk herum, wer denn nun der Kollege mit dem korrekten Namen und der richtigen Adresse ist. Wenn der jeweilige Rechner sich nicht zuständig fühlt, wird die Anfrage an den nächsten Rechner weitergeleitet, bis sie an der richtigen Stelle angekommen ist.

Richtig haarig wird diese Angelegenheit, wenn der zuständige Rechner zwar zur Arbeitsgruppe gehört, aber nicht im selben Netzwerk zu Hause ist wie die anderen. Broadcast-Pakete können nicht geroutet werden. Dann müssen Sie auf die »lmhosts«-Datei zurückgreifen und haben praktisch den gleichen Aufwand wie anno domini 1969, als man die Hostnamen mit einer »hosts«-Datei auf jedem Rechner verwaltete.

Mit WINS ist das anders. Der Client geht nicht mit seiner Anfrage hausieren, sondern wendet sich vertrauensvoll an seinen Server und klärt mit ihm die Kompetenzen, bevor er versucht, eine Verbindung nach irgendwo aufzubauen.

Samba bietet diesen Service auch. Zusätzlich kann Samba, sofern es die Daten nicht selbst zur Verfügung hat, den DNS-Dienst nach den korrekten Adressen fragen. Es macht auch aus diesem Grund Sinn, einen Nameserver im lokalen Netz zu betreiben. Dann müssen Sie grundsätzlich nur noch eine Datenbank pflegen.

Damit das Ganze funktioniert, müssen alle Rechner, die sich kennen sollen, den gleichen WINS-Server verwenden.

Ein Master-Browser

Das NetBIOS-Netz wird in logische Arbeitsgruppen unterteilt. Jeder Client und jeder Server gehört immer zu genau einer Arbeitsgruppe. In den Konfigurationsbeispielen heißt sie unter Samba »Workgroup«, zu Deutsch »Arbeitsgruppe«. Dieser Parameter kann selbstverständlich geändert werden.

Wer Master-Browser in der Arbeitsgruppe wird, wird unter den elektronischen Mitgliedern der Gruppe abgestimmt. Dabei können in Samba Präferenzen über den *prefered master*-Parameter gesetzt und so die Wahl einer bestimmten Maschine erzwungen werden. Zusätzlich müssen Sie in so einem Fall ein »OS-Level« definieren. Eine solche Zahl ist ein einfacher Integer-Wert. Je höher diese Zahl ist, desto mehr Stimmen hat der Rechner und desto höher ist die Wahrscheinlichkeit, daß besagtes Mitglied zum Kanzler des Rechnervolkes wird. Möglich sind Werte von 0 bis 255.

Aufgabe des Master-Browsers ist es, Informationen über alle im NT-Netz zur Verfügung stehenden Dienste einzusammeln und sie den anderen Kollegen im Netz bekannt zu machen.

Diese Master Browser sind wiederum hierarchisch unterteilt: es gibt einen Local Master Browser, einen Domain Master Browser und einen Backup Master Browser. Wenn das Netz größer wird, nimmt die Anzahl der Server zu, die im Netz etwas anzubieten haben. Ob es sich dabei um Daten (Shares) oder Dienste wie das Drucken handelt, ist bei dieser Betrachtung nicht relevant. In der lokalen Subdomain erledigt der Local Master Browser diese Aufgabe.

Der Domain Master Browser tut prinzipiell das gleiche. Der Unterschied ist, daß er diese Arbeit auch über die Grenzen der einzelnen Subdomain hinweg erledigen kann.

Der Backup Master Browser dient nur einem Zweck (seine Daten entsprechen denen des Domain Master Browsers): er wird nur aktiv, wenn der Domain Master ausgefallen ist.

Damit Sie einen Überblick über diese Hierarchie bekommen, hier eine Tabelle mit den Kennungen des OS-Levels:

OS-Level	Betriebssystem
1	Windows for Workgroups
1	Windows 95
1	Windows98/Millenium
16	Windows NT 3.51 Workstation
17	Windows NT4 Workstation
32	Windows NT 3.51 Server
33	Windows NT4 Server
Nach Auswahl	Ein Samba-Server ...

Tabelle 9.1: Die Hierarchie des OS-Level im Windows-Netzwerk. Theoretisch sind Angaben zwischen 0 und 255 möglich. Die Voreinstellung bei Samba 2.2.5 ist 20

Wenn über den Domain Master Browser abgestimmt werden soll und die Wahl aufgrund des OS-Levels nicht durchgeführt werden kann, kommen noch weitere Kriterien bei der Entscheidungsfindung dazu. Bei gleichem Level gewinnt in der Regel der Server die Wahl, dem am längsten online ist. Wenn das auch noch nicht reicht, wird bei der Abstimmung eine lexikalische Bewertung durchgeführt.

Um diese Angelegenheit richtig kompliziert und unübersichtlich zu machen, wird diese Abstimmung immer wieder durchgeführt, wenn:

- Sich ein neuer und als Master-Browser geeigneter Kandidat im Netz anmeldet,
- wenn ein Client einen Browser zu kontaktieren versucht,
- oder wenn ein Backup-Browser seine Browseliste auffrischen will und keinen Domain Master Browser vorfindet.

Jeder Server, der neu im Netz ist, fragt zuerst einmal herum, wer der aktuelle Browser ist. Dazu wird ein sogenanntes »Server Announcement Package«, ein simpler Broadcast, mit dem nach den Zuständigkeiten gefragt wird, im Netz herumgeschickt. Der gerade aktuelle Domain Master Browser bekommt den Broadcast, setzt den Namen des Rechners auf seine Browse-Liste und teilt ihm außerdem mit, wer alles als Backup Browser arbeitet. Ist dieser Rechner ein potentieller Master Browser, wird neu abgestimmt.

Die Unterschiede in Windows2000/XP

Die Hierarchie in Windows-2000-Netzwerken unterscheidet sich etwas von der in den klassischen NT-Netzwerken. Die Rolle eines Primary Domain Controllers übernimmt eine Maschine, die dort nur noch Domain Controller heißt. Die Rolle des Backup Domain Controllers übernimmt ein anderer Server in der Domain, der nur Mitglied ist, und der Rest bleibt wie gehabt.

9.2.4 Die Freigabe von Netzwerklaufwerken

Sie können Netzlaufwerke nach mehreren Parametern freigeben: Es gibt die Freigabe nach Netzwerken oder Arbeitsgruppen, den Weg, bestimmte Verzeichnisse nur für bestimmte User freizugeben, den Weg, alle Verzeichnisse in einem bestimmten Teil des Baums für alle les- und/oder beschreibbar zu machen, oder einfach den Zugriff über die Hostnamen zu regeln.

Wenn Sie Dateisysteme über Samba für andere Benutzer freigeben wollen, muß in einem einfachen Netz dem Server der Name des Benutzers, der auf dem Server seine Daten bearbeiten soll, bekannt sein. Sie müssen in einem einfachen Netzwerk also zwei Benutzer mit dem gleichen Namen verwalten: den auf dem Windows-System und den auf dem Server als Systembenutzer. Die Ausnahme ist, wenn Sie das Einloggen zentralisieren und mit einer NT-Domain arbeiten.

9.2.5 Netzwerkdrucker

Wenn Sie einen Drucker auf Ihrem NetBSD-Server betreiben, auf den die Windows-Clients zugreifen sollen, müssen Sie ihn für die Arbeitsgruppe freigeben. Das bezieht sich aber nur auf Drucker, die mit Windows 9x laufen. Für Windows NT, 2000 und XP gibt es den im Buch beschriebenen Unix-LPD-Client, der das SMB-Protokoll zumindest für den Druckservice überflüssig macht.

Wenn Sie nur den Druckdienst der NetBSD-Maschine nutzen wollen, sollten Sie das Verfahren mit dem Programm *pmlpr* in Kombination mit dem LPD anwenden, wenn sie mit Windows 9x-/Millenium-Systemen zu tun haben. Das ist in so einem Fall effizienter, weil ein Dienst weniger auf der NetBSD-Maschine läuft und der Arbeitsaufwand sowohl für den Server als auch für Sie selbst dadurch deutlich geringer wird. Wenn Sie allerdings auch die Filesharing-Dienste anbieten wollen (deshalb lesen Sie dieses Kapitel wahrscheinlich), sieht es anders aus. Dann können Sie das mit den Parametern *load printers=yes* und *printing = "bsd"* erledigen.

9.2.6 Die Installation

Sie können, wie bei allem, was mit NetBSD zu tun hat, die Software auf verschiedenen Wegen installieren. Zu bevorzugen ist im Regelfall der Weg aus dem Package-Baum, der, sofern Sie die Entwicklerpakete mitinstalliert haben und Sie den Rechner dafür vorbereitet haben, der einfachste ist. In »/usr/pkgsrc/samba« rufen Sie den folgenden Befehl auf:

```
# make install
```

Wenn das getan ist, wird der Rechner sich die notwendigen Daten aus dem Internet besorgen, sie auspacken, kompilieren und an der richtigen Stelle installieren.

Alternativ geht die Installation natürlich auch von der CD mit einem *pkg_add samba**.

Dann müssen Sie den Diensten *netbios-ssn* (das ist der Daemon *smbd*) und *netbios-ns* (und dies der Daemon *nmbd*) jeweils einen Port zuweisen. Das sind die Ports 137/udp für *netbios-ns* und 139/tcp für *netbios-ssn*.

Eingetragen wird das in der Datei »/etc/services«.

Sie können (wie bei allem, das mit NetBSD zu tun hat) den Samba-Server auf drei Arten starten. Die erste Alternative ist der Weg über den *inetd*, der manchmal auch der einfachste ist.

Das geht mit diesen Einträgen in der »/etc/inetd.conf«:

```
netbios-ssn stream tcp nowait root /usr/sbin/smbd -dl <Domain> -n <Server>
netbios-ns degram udp wait root /usr/sbin/nmbd nmbd -dl
```

Der Nachteil dieses Verfahrens ist, daß der Verbindungsaufbau deutlich länger dauert als wenn Samba über das rc-System als Daemon gestartet wird. Dann befindet sich der Server bereits im Hintergrund im Arbeitsspeicher und wartet darauf, an die Arbeit gehen zu dürfen.

Die mitgelieferte Datei zum Starten dieses Dienstes über das init-System sieht so aus:

```
if [ ! -d /usr/bin ]
then                    # /usr not mounted
        exit
fi

killproc() {            # kill the named process(es)
        pid=`/bin/ps -e |
             /usr/bin/grep -w $1 |
             /usr/bin/sed -e 's/^  *//' -e 's/ .*//'`
        [ "$pid" != "" ] && kill $pid
}

# Der Start-/Stop-Prozess

case "$1" in

'start')
#
# Hier steht drin, wo Samba liegt und mit welchen Parametern er gestartet
werden soll.
```

```
# Wie Sie sehen, ist das Skript bereits an NetBSD angepasst.
#
   /usr/pkg/sbin/smbd -D -s/etc/samba/smb.conf
   /usr/pkg/sbin/nmbd -D -l/etc/samba/log -s/etc/samba/smb.conf
   ;;
'stop')
   killproc nmbd
   killproc smbd
   ;;
'restart')
stop
echo "restarting samba"
start
;;
*)
   echo "Usage: /etc/init.d/samba { start | stop | restart}"
   ;;
esac
```

Das ist ein komplizierter Weg, um Samba zu starten. Sie haben auch die Möglichkeit, den Job vom Startskript »/etc/rc.local« aus erledigen zu lassen und sich die eigentlich überflüssige Variante über die »rc.d« zu sparen. Das tun Sie mit diesen Einträgen, die Sie einfach an das Ende des »rc.local«-Skripts anhängen, das es ja genau für diesen Zweck gibt:

```
/usr/pkg/sbin/nmbd -D
/usr/pkg/sbin/smbd -D
/usr/pkg/sbin/smbd start
```

Wenn Sie das getan haben und ein Windows-Rechner (diesmal ein beliebiger) bei Ihnen konfiguriert und lauffähig mit Netzwerkanbindung in einer dunklen Ecke herumsteht, sollten Sie versuchen, Samba zu starten. Das machen Sie mit:

```
# /usr/pkg/sbin/nmbd -D
```

... und einem

```
# /usr/pkg/sbin/smbd -D
# /usr/pkg/sbin/smbd start
```

um sich dann von der Windows-Maschine aus versuchsweise mit einem Verzeichnis auf dem NetBSD-Server zu verbinden. Wenn die Windows-Maschine richtig konfiguriert ist und Sie auf dem NetBSD-Rechner ein Benutzerkonto besitzen, sollten Sie bereits auf die Daten in den »/home«- Verzeichnissen zugreifen können, und das, ohne vorerst eine einzige Zeile in der Konfigurationsdatei zu ändern oder diese überhaupt gesehen zu

haben. Außerdem sollte der Drucker bereits funktionieren, wenn er auf der NetBSD-Maschine läuft, richtig konfiguriert ist und Sie ihn auf den Windows-Clients entsprechend, diesmal über Samba, konfiguriert haben.

9.2.7 Samba auf die Schnelle

Nach der Installation des Samba-Servers befinden sich außer den oben angesprochenen Daemons *nmbd* und *smbd* einige weitere Programme und eine wichtige Konfigurationsdatei im System. Die Konfigurationsdatei besitzt den Namen »smbd.conf« und befindet sich in »/usr/pkg/etc/samba«.

Diese Lage der Datei ist aus NetBSD-Sicht sinnig, aus Sicht einiger Konfigurations-Frontends ist sie es weniger. Sie sollten sich deshalb überlegen, ob sie nicht einen symbolischen Link dieser Datei nach »/etc« legen. Sie müssen das nicht, aber einige Konfigurationstools erwarten die Datei »/etc/smb.conf« und verlangen, wenn sie die Datei nicht unter diesem Namen finden, daß man den kompletten Pfad eingibt oder sich einmal durch den Verzeichnisbaum bis nach »/usr/pkg/etc/samba« durchklickt.

In der Voreinstellung – es soll hier ja um eine Ruck-zuck-Lösung gehen, ist Samba so eingestellt, daß die Passworte in einer eigenen Passwortdatei lokal liegen. Das Passwort für jeden Samba-User muß also angelegt sein. Dabei muß der Samba-User selbst als Unix-User im System vorhanden sein.

Um die User für Samba anzulegen, sollten Sie also vor der Inbetriebnahme des Servers die User definieren. Dazu rufen Sie das Hilfsprogramm *smbpasswd* mit dem Parameter *-a* für »add« und dem Namen des jeweiligen Users auf:

```
smbpasswd -a username
```

Es folgt bei jedem Aufruf die übliche Abfrage des Passworts und dessen Bestätigung.

Damit ist der erste Teil geschafft.

Im nächsten Schritt wenden wir uns zwangläufig der Konfigurationsdatei selbst zu. Sie muß in einen beliebigen Editor geladen werden.

Die Entscheidung, wie weiter verfahren wird hängt davon ab, ob eine Windows-Maschine in ein bestehendes Unix-Netz oder ein NetBSD-Rechner in ein bestehendes Windows-Netz als Server integriert werden soll.

Gehen wir vom zweiten Fall aus, er ist verständlicher und schneller aufgesetzt. Im Regelfall sieht dieses Szenario so aus, daß ein NT-Server vorhanden ist, der die Windows-Arbeitsgruppe verwaltet. Der Unix-Rechner soll jetzt so eingerichtet werden, daß Verzeichnisse auf der Platte den Windows-Usern automatisch zur Verfügung gestellt werden. Zum Schluß soll es dann so sein, daß im Netzwerk-Dialog dieser Rechner zu sehen ist (Abb. 9.2).

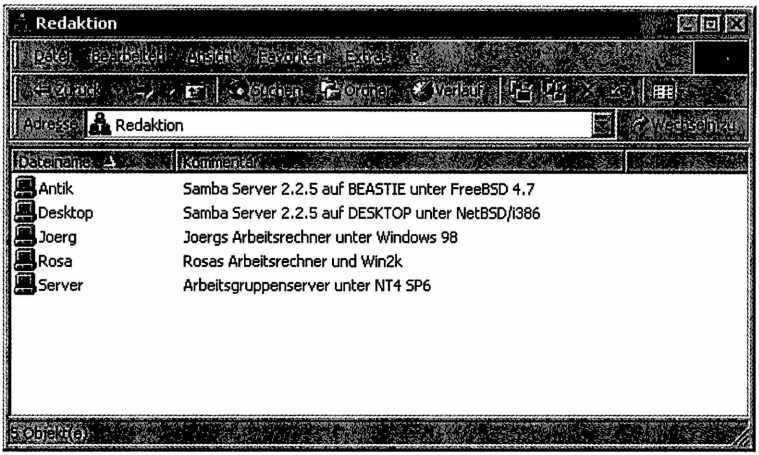


Abb. 9.2: Ein Windows-Netzwerk mit einem NT-Server, zwei Clients und einem NetBSD- und einem FreeBSD-Rechner. Am ersten Rechner ist zu sehen, daß der Hostname und der Name in der Arbeitsgruppe keineswegs übereinstimmen müssen.

Die Voraussetzungen für die Einbindung des BSD-Rechners in das Windows-Netz sind trivial.

Die Samba-Konfigurationsdatei besitzt äquivalent zu einer Windows-Ini-Datei mehrere Sektionen. Entscheidend ist die Sektion *global*. Alle Eintragungen hier gelten – wie der Name schon sagt – global. Zu definieren ist hier der Name der Arbeitsgruppe, voreingestellt ist hier *WORKGROUP*, und ob eine Verschlüsselung der Passworte stattfinden soll. Diese beiden Eintragungen müssen stimmen.

In einem bestehenden Windows-Netz muß als Name der Arbeitsgruppe der Name angegeben werden, wie er auf dem NT-Server und den anderen Windows-Maschinen definiert ist. Wird hier ein anderer Name angegeben, erscheint der Samba-Server nicht im Netzwerk-Dialog. Ab Windows 95 OSR und Windows NT Service Pack 4 verschlüsseln alle Windows-Maschinen die Passworte. Wenn keine älteren Maschinen anzutreffen sind, muß die Verschlüsselung eingeschaltet werden. Es gibt Workarounds, wenn sich auch ältere Maschinen (Windows 95, Windows für Workgroups) im Netz befinden. Sie finden die Hinweise zum Abschalten der Passwortverschlüsselung in der Manual-Seite der »smb.conf«.

Die globale Sektion der »smb.conf« sollte im Normalfall, also wenn sich keine alten Maschinen ohne Passwortverschlüsselung in der Arbeitsgruppe befinden, ungefähr wie folgt aussehen:

```
[global]
  workgroup         = REDAKTION       ; Der Name der Arbeitsgruppe, anpassen!
  server string     = Samba Server %v auf DESKTOP unter NetBSD/i386 1.6
  encrypt passwords = Yes
  update encrypted  = Yes
  security=user
  log file          = /usr/pkg/samba/var/log.%m
  max log size      = 50
  preferred master = False
  local master      = no
  domain master     = False
  dns proxy         = No
  load printers     = no
```

Soll ein vorhandener eingerichteter NT-Server weiterhin über die Passworte wachen, kann die Datei auch so aussehen:

```
[global]
  workgroup=REDAKTION            ; Der Name der Arbeitsgruppe, anpassen!
  server string=Samba Server %v auf DESKTOP unter NetBSD/i386 1.6
  encrypt passwords=Yes
  update encrypted=Yes
  security=server                   ; sehr bequem!
  password server=192.168.10.10     ; .. oder der Name des Servers. Anpassen!
  log file=/usr/pkg/samba/var/log.%m
  max log size=50
  preferred master=FALSE
  local master=No
  domain master=FALSE
  dns proxy=No
  load printers=no
```

Sie sehen an den unterschiedlichen Schreibweisen, daß Samba bei der Interpretation der Eintragungen sehr flexibel ist. Groß- und Kleinschreibung wird ignoriert, außerdem ist es egal, ob vor und hinter den zuweisenden Gleichheitszeichen Leerzeichen oder Tabulatoren folgen oder nicht.

Einige der Einstellungen hier entsprechen den Voreinstellungen und sollen nur demonstrieren, daß es sich hier im Sinne des Windows-Netzes um einen reinen Client handelt. Soll der Name der Maschine im Windows-Netz von dem Unix-Namen abweichen, hat noch die Angabe des *netbios name* zu erfolgen. Dies ist aber schon wieder Kosmetik.

Nun sollte noch die Sektion *homes* kontrolliert werden. Im Gegensatz zum Windows-Netz kann man Samba dazu bringen, daß das Heimatverzeichnis nur für den User selbst sichbar ist. Dazu dient die Sektion *homes*. Sie wird im Windows-Netz den Share-Namen des jeweiligen Users umgesetzt:

```
[homes]
  comment = Heimatverzeichnis von %u
  valid users = %S
  read only = No
  browseable = No
```

Um für den Datenaustausch der User untereinander auch noch ein gemeinsames Verzeichnis zur Verfügung zu haben, sollte man zu den bisherigen Verzeichnissen auch noch ein *Public* freischalten. Idealerweise legt man es als */home/samba* an und vergibt diesem Verzeichnis auf Unix-Ebene die Attribute 777, damit jeder Anwender die Daten lesen und schreiben kann:

```
[pub]
  comment = Oeffentliches Verzeichnis
  path = /home/samba
  read only = No
  create mask = 0666
  directory mask = 0777
  guest ok = Yes
```

Warum die Unix-Attribute diese Bedeutung haben, wird später in diesem Kapitel erläutert. Für den Moment soll genügen, daß es so sein muß.

Besitzt die Datei diese Eintragungen – alles andere sollte entweder mit Strichpunkten oder einem Gatter-Zeichen auskommentiert sein – muß geprüft werden, ob die Angaben auch syntaktisch richtig sind. Dazu wird die Datei gespeichert, der Editor verlassen und das Prüfprogramm *testparm* gestartet:

```
[/usr/pkg/etc/samba]$testparm
Load smb config files from /usr/pkg/etc/samba/smb.conf
Processing section "[homes]"
Processing section "[pub]"
Loaded services file OK.
Press enter to see a dump of your service definitions
```

Taucht hier eine Fehlermeldung auf, daß »/usr/pkg/samba/var/« nicht vorhanden ist (der Pfad kann frei gewählt werden), ist ein anderes Verzeichnis für die Logs zu wählen oder das Verzeichnis anzulegen.

Es ist unbedingt darauf zu achten, daß der Test fehlerfrei ist. Nur ein Fehler – und klingt er noch so unbedeutend – heißt, daß es nicht klappen wird!

Hat *testparm* die Fehlerfreiheit bestätigt, müssen die beiden Server *smbd* und *nmbd* hochgefahren und *smbd start* aufgerufen werden. In den folgenden Zeilen ist der Pfad jeweils angegeben, da sich »/usr/pkg/sbin« im Suchpfad von *root* befindet, das ist aber eigentlich nicht nötig:

```
# /usr/pkg/sbin/nmbd -D
# /usr/pkg/sbin/smbd -D
# /usr/pkg/sbin/nmbd start
```

Damit ist der Server testweise aktiviert. Ein Aufruf von *smbclient* verrät, ob die Shares freigegeben sind:

```
# smbclient -L localhost -U joerg
added interface ip=192.168.10.16 bcast=192.168.10.255 nmask=255.255.255.0
Password:
Domain=[REDAKTION] OS=[Unix] Server=[Samba 2.2.5]

        Sharename       Type      Comment
        ---------       ----      -------
        pub             Disk      Oeffentliches Verzeichnis
        IPC$            IPC       IPC Service (Samba Server 2.2.5 auf DESKTOP
unter NetBSD/i386 1.6)
        ADMIN$          Disk      IPC Service (Samba Server 2.2.5 auf DESKTOP
unter NetBSD/i386 1.6)
        joerg           Disk      Heimatverzeichnis von joerg

        Server               Comment
        ---------            -------
        DESKTOP              Samba Server 2.2.5 auf DESKTOP unter NetBSD/i386
        SERVER               Arbeitsgruppenserver unter NT4 SP6

        Workgroup            Master
        ---------            -------
        REDAKTION            SERVER
```

Die Angaben sind vielversprechend. *IPC$* und *ADMIN$* sind Verwaltungs-Shares und ansonsten in diesem Zusammenhang bedeutungslos, der Master-Server wurde erkannt, die Shares für das Home-Verzeichnis und Public sind freigegeben.

Der Server läuft!

Sie können jetzt versuchen, von einer Windows-Maschine aus auf den Rechner zuzugreifen. Samba-Clients gibt es übrigens nicht nur unter Windows. OS/2 Warp unterstützt ab der Version 3 Connect das Zusammenspiel mit dieser Art der Netzwerkverbindung perfekt. Außerdem enthalten FreeBSD und Linux *mount*-Befehle für Samba-Shares. Unter FreeBSD kann ein NT-Rechner oder ein Samba-Share folgendermaßen in die »/etc/fstab« eingetragen werden:

```
//joerg@desktop/pub /mnt/desktop smbfs rw,noauto  0  0
```

Es handelt sich um das oben freigeschaltete Public-Verzeichnis. NetBSD besitzt kein solches Client-Programm. Wie man von einer NetBSD-Maschine auf Samba-Shares zugreift, ist in Kapitel 9.3 beschrieben.

Eines ist im Zusammenhang mit Samba noch wichtig: Wenn Sie eine Verwaltung wie in dieser Kurzanleitung beschrieben vornehmen wollen, ist es wichtig, daß die Datei »/etc/hosts« gepflegt ist und wenigstens auch der eigene Name der Maschine mit der IP-Adresse eingetragen ist! Ist das nicht der Fall, funktioniert nicht einmal der lokale Test mit *smbclient*.

9.2.8 Samba verstehen

Eine einfache Konfiguration, in der ein Samba-Server seine Daten anderen Systemen im Netz zur Verfügung stellt, kann aussehen wie diese hier. Die »home«-Verzeichnisse werden so freigegeben, daß sich nur Benutzer des NetBSD-Servers auch von einer Windows-Maschine aus anmelden können. Zusätzlich gibt es in diesem Beispiel noch ein Verzeichnis, das von allen Benutzern verwendet wird, um Dateidownloads aus dem Internet lokal weiter zu verteilen, was ja manchmal auch Sinn macht. Unterteilt ist die Konfiguration grundsätzlich in zwei Sektionen: *Global* mit den allgemeinen Parametern und nach Diensten beziehungsweise Benutzern. In der globalen Sektion können nicht nur »allgemeine« Eintragungen zum Server gemacht werden. Es werden auch Eintragungen ausgewertet, die für Shares – dann aber für alle – gelten. Recht praktisch ist es, wenn man beispielsweise eine *directory mask* = 0777 angibt. Das bedeutet, daß unabhängig von den Benutzerrechten der Dateien in den Verzeichnissen dafür gesorgt wird, daß der Windows-Anwender in andere von Windows-Anwendern angelegte Verzeichnisse wechseln kann. Der normale Windows-User wird auf das Verhalten, daß ein Verzeichnis zwar angezeigt, aber nicht »betretbar« ist, eher mit Unverständnis reagieren.

In die globale Sektion werden alle Parameter eingetragen, die für alle Shares und Dienste zutreffen sollen:

```
[global]
  security       = share
  server string  =SAMBA %v auf NetBSD 1.6/i386
  strict locking = yes
  log level      = 1
  max log size   = 1000
  guest ok       = no
  workgroup      = Arbeitsgruppe
  netbios name   = Hostname
```

Die restlichen Konfigurationsdaten definieren die Shares selbst. Dabei ist »homes« ein Spezialfall:

```
[homes]
create mask = 700
comment=Heimatverzeichnis von %u
writeable = yes
browseable=no
; read only=no
```

Während bei normalen Shares immer der Name für den Eintrag im Windows-Netzwerk steht, ist das bei diesem Share nicht der Fall. Es ist hier auch zu sehen, daß kein Pfad angegeben ist. Das ist prinzipiell mit einem

```
path=/home/%U
```

zwar möglich, aber grundsätzlich überflüssig. *ready only = no* ist auskommentiert, da es die gleiche Wirkung wie *writeable=yes* hat.

Der Eintrag *{homes}* dient dazu, daß jeder Anwender im Netzwerk immer sein eigenes Heimatverzeichnis unter seinem Namen angezeigt bekommt. Samba ermittelt beim Einloggen das Heimatverzeichnis des Users und zeigt bei der Anwahl dessen Inhalt an. Die Heimatverzeichnisse anderer Benutzer bekommt man praktischerweise gar nicht erst angezeigt. Dafür sorgt der Eintrag *browseable = no*.

Bei anderen Shares entspricht der Name der entsprechenden Sektion immer dem Freigabenamen:

```
[downloads]
path = /home/downloads
public = yes
create mask = 0644
writeable = yes
directory mask = 0777
```

Dieses Verzeichnis ist für jeden Benutzer sichtbar. Dateien, die angelegt werden, erhalten die Unix-Dateiattribute 644, sind also für den Eigentümer les- und schreibbar, andere Anwender können sie nur lesen und damit auch nicht versehentlich überschreiben. In die über Samba angelegten Verzeichnisse kann jedoch jeder wechseln.

Es macht überhaupt keinen Sinn, Dateien mit dem Ausführen-Flag zu versehen (also beispielsweise mit 0755, da es sich bei den abgelegten Dateien im Regelfall um Windows-Daten handelt. Sie sind wohl sehr selten unter Unix ausführbar).

Falls ein Rechner ein oder mehrere CD-Laufwerke besitzt, wäre es praktisch, wenn die Anwender auf diese Laufwerke zugreifen können. Dazu dient der Share *cdrom*. Hier eine einfache Variante:

```
[cdrom]
path = /mnt/cdrom
public = yes
writeable = no
```

Der Haken ist, daß das Laufwerk gemountet sein muß.

Es gibt deshalb unter Samba die Möglichkeit, für Shares eine Aktion zu definieren. Sie können Samba – insbesondere bei CD-Laufwerken macht das Sinn – auch noch den Auftrag erteilen, vor und nach dem Einloggen eines Benutzers bestimmte Aktionen durchzuführen. Es ist beispielsweise praktisch, daß, wenn ein Benutzer auf *{cdrom}* zugreift, eine CD auch eingebunden ist. Nachdem keine Zugriffe mehr erfolgen, sollte automatisch ein Unmount durchgeführt werden. Dafür sind die beiden folgenden Zeilen in [cdrom] zuständig:

```
root preexec = mount /mnt/cdrom
root postexec = umount /mnt/cdrom
```

Der Mount wird dabei als User *root* durchgeführt. Ist keine CD-ROM eingelegt, stört das nicht. Das Verzeichnis wird dann ohne weitere Daten angezeigt. Da das Schreiben mit *writeable = no* unterbunden ist, können auch keine Daten in den Mountpunkt selbst geschrieben werden, die beim Mount einer CD-ROM versteckt sind.

Auf die gleiche Art und Weise läßt sich auch ein Diskettenlaufwerk freigeben:

```
[floppy]
path = /mnt/floppy
public = yes
writeable = no
root preexec = mount /mnt/floppy
root postexec = umount /mnt/floppy
```

Die Definition von *homes* ist eine recht praktische Angelegenheit, da der Samba-Server dafür sorgt, daß jedem Anwender automatisch sein Heimatverzeichnis zugewiesen wird. In manchen Fällen ist das aber nicht erwünscht. Unter Umständen sollen nicht alle Anwender Heimatverzeichnisse besitzen, andere sollen auf die Maschine Zugriffe haben und nicht nur als Gast, sondern als regulärer Anwender aber nur auf bestimmte Verzeichnisse. Dann ist es genauso möglich, Heimatverzeichnisse wie normale Verzeichnisse zu definieren, aber nur bestimmten Anwendern oder Anwendergruppen den Zugriff zu erlauben. Dann wäre für die Heimatverzeichnisse die folgende Definition zu wählen:

```
[Ihr Name]
path=/home/[Ihr Nutzername]
public = no
writeable = yes
valid users = [sie]
```

Wenn der Benutzer *Fritz* ein Projekt betreut, auf das auch *werner* und *anke* zugreifen sollen, wird sein Heimatverzeichnis folgendermaßen freigeschaltet:

```
[Fritz]
path=/home/fritz
map archive = yes
public = no
writeable = yes
valid users = fritz, werner, anke
```

Variable	Bedeutung
%u	Der Benutzername.
%g	Der primäre Gruppenname von %u.
%U	Der Benutzername der Sitzung. Das ist der Name, unter dem sich ein Samba- Benutzer auf der Maschine einloggt.
%G	Der Gruppenname von %U.
%H	Das Home-Verzeichnis von %U.
%v	Gibt den Versionsnamen des Samba-Servers aus.
%h	Der Unix-Name, also der Internet-Name des Servers.
%m	Der NetBIOS-Maschinenname des Clients.
%L	Der aktuelle NetBIOS-Name des Clients. Eine Windows-Maschine kann mehrere davon besitzen.
%M	Der Internet-Name der Client-Maschine.
%a	Damit kann ausgegeben werden, mit welcher Systemarchitektur die Client-Maschine läuft. Identifiziert werden können Samba, Windows3.x, Windows9x und NT.
%I	Gibt die IP-Adresse der Client-Maschine aus.
%T	Aktuelles Datum und Uhrzeit.
%j	Jobname.
%S	Name der Spooldatei.

Tabelle 9.2: Die wichtigsten Platzhalter in der »smb.conf«

9.2.9 Samba und seine Benutzer

Nach der Besichtigungstour durch diese wirklich sehr simple Konfigurationsvariante sind wir schon beim ersten wichtigen Thema zu Samba angekommen: der Benutzerverwaltung.

Abgesehen von den unter Samba beim Security-Level *share* möglichen Gast-Benutzern müssen alle Systembenutzer, also auch die, die über Samba auf den Server zugreifen sollen, über ein Standard-Benutzerkonto auf der NetBSD-Maschine verfügen.

Es ist unter Umständen auch empfehlenswert, alle Samba-Benutzer in einer eigenen Gruppe zusammenzufassen. Das vereinfacht den Überblick über diese Benutzer und darüber, was sie dürfen und was nicht.

Dann haben Sie meistens eine größere Gruppe von Nutzern, die nur die Drucker und die Dateidienste nutzen sollen, sich aber aus Sicherheitsgründen nicht via Telnet oder *ssh* mit dem Samba-Server verbinden sollen.

Zu guter Letzt gibt es auch noch die Gruppe von Benutzern, die sich auch direkt mit der NetBSD-Maschine verbinden können soll, um darauf zu arbeiten.

Der Lösungsansatz für die Benutzer, die nur Samba nutzen sollen, sieht so aus:

Sperren Sie für diese User den Zugang auf die Shell, indem Sie in »/etc/passwd« diesen Benutzern die Shell */sbin/nologin* zuweisen. Der Server weiß dann, daß dieser Benutzer keinen Shellzugang bekommt, wenn er einen Loginversuch startet.

Wenn Sie auf ihrer Samba-Maschine auch einen FTP-Server betreiben, sollten Sie diese Benutzer auch vom FTP-Dienst aussperren. Das machen Sie in der Datei »/etc/ftpusers«, indem Sie den Namen des Benutzers eintragen und in derselben Zeile ein *deny* hinter diesen Namen setzen. Vier Namen sind in der Datei bereits aufgeführt, es sollte also nicht zu schwierig werden.

Beachten müssen Sie, daß die Passwortsysteme von Windows- und Unix-Systemen (also auch von NetBSD) nicht miteinander kompatibel sind. Wenn Sie mit verschlüsselten Paßwörtern arbeiten wollen, müssen Sie die Windows-Paßwörter der Samba-Benutzer separat in einer Datei namens »smbpasswd« pflegen. Diese Datei finden Sie unter »/usr/pkg/etc/samba«.

Mit folgendem geben Sie Samba einen neuen Benutzer bekannt:

```
# /usr/pkg/bin/smbpasswd -a [Username]
```

Das müssen Sie allerdings nur tun, wenn Sie mit verschlüsselten Paßwörtern arbeiten und wenn die Paßwörter lokal gehalten werden.

Damit sind wir dann auch schon beim zweiten Thema, dem Security-Level.

Hier gibt es vier Varianten: einmal den *share*-Parameter, dann den *user*-Parameter und zu guter Letzt den *server*- und den *domain*-Parameter.

Der Security-Level entscheidet darüber, um welche Art von Windows-Netzwerk es sich handelt.

Was Sie außerdem bedenken sollten:

Wenn ein einfaches Windows-Netzwerk mit der Security-Einstellung *share* vorliegt, sollte Ihnen bewußt sein, daß zumindest bei den Windows-9x/ME-Clients keine Benutzerverwaltung stattfindet, weil diese Systeme keine unautorisierten Benutzer kennen. Das sieht so aus: Wenn sich jemand auf einem solchen Rechner einloggt, legt eine derartige Maschine kurzerhand ein Benutzerkonto an. Ein derartiger Benutzer kann auf alle Shares und Drucker im Netz zugreifen, die öffentlich sind.

Das User-Level ist sicherer. Es werden nur Anmeldungen von registrierten Benutzern akzeptiert. Wenn das Login abgebrochen wird, kann auch nicht auf das Netzwerk zugegriffen werden, was anders bei den Windows-Maschinen möglich wäre.

Exportiert man ein Verzeichnis über Samba, gibt es natürlich auch Lese-/Schreib- und Ausführungsberechtigungen, die zu vergeben sind.

Setzt man den Parameter *writeable* auf *yes*, heißt das zunächst nur, daß der Eigentümer des Verzeichnisses in dieses Verzeichnis schreiben soll. Das heißt aber nicht, daß er das auch darf. Geregelt wird der eigentliche Zugriff auf diese Daten immer noch nach den Unix-Standards. Das bedeutet, daß die Unix-Berechtigungen vor denen in der »smb.conf« Vorrang haben und in der Hierarchie immer über dem stehen, was in der Samba-Konfiguration steht. Vergessen Sie das bitte nicht; besonders, wenn Sie zum ersten Mal einen solchen Server aufsetzen, sorgt das bei öffentlichen Verzeichnissen bisweilen für Konfusion.

Der Regelfall für eine solche Maske für ein Benutzerverzeichnis wird 0755 sein, bei Gästen eher 0744. Die Syntax in der »smb.conf« dafür ist

```
create mask = 0755
```

Ist das Security-Level auf *share* gesetzt und ist ein Gastbenutzer auf dem NetBSD-System eingerichtet, können Sie zwei Schienen fahren: Sie können einmal die Freigabe so einrichten wie oben und zusätzlich die Zugriffe für Verzeichnisse und Dateien, die dieser Gast anlegt, separat regeln. Das machen Sie mit:

```
directory mask = 0755
```

Unix unterscheidet bekanntlich bei den Rechten zwischen Lesen, Schreiben und Ausführen (oder auch Hineinwechseln). Unter Windows funktioniert das etwas anders: Dort gibt es die Attribute read-only, System, Archiv und Versteckt. Das muß natürlich auch auf einem Samba-Server übersetzt werden. Die ersten beiden Bits (Lesen und Schreiben) werden als read-only übersetzt; das Ausführungsbit steht für »Archiv«; das Bit der Gruppe steht für »System« und das für »andere« stellt dar, ob diese Datei versteckt werden soll.

9.2.10 Die Security-Einstellungen

Mit den Security-Einstellungen wird geregelt, in welcher Form auf den Samba-Server zugegriffen werden soll. Es gibt insgesamt vier, die davon abhängen, wie der Samba-Server funktionieren soll.

share:

Ein klassisches Peer-to-Peer-Netzwerk, wie es eine Arbeitsgruppe aus mehreren Windows-Rechnern (diesmal im Ansatz Nicht-NT) darstellt, benutzt im Regelfall immer den Parameter *share*. Das heißt, daß die Zugriffe auf Freigabe-Ebene geregelt werden. Wenn man auf eine Maschine mit diesem Security-Level zugreifen will, wird bei jedem Zugriff auf eine einzelne Freigabe die Berechtigung des Clients überprüft. In diesem Security-Level muß man sich nicht bei jedem Zugriff auf den Server mit Namen und Passwort anmelden, um auf die Shares zugreifen zu können. Das bedeutet allerdings nicht, daß der User, der auf Daten zugreifen will, dem Server nicht bekannt sein muß – auf dem NetBSD-Server braucht der Benutzer trotzdem ein Konto. Nur wenn auf einen Share zugegriffen wird, wird nach dem Passwort des Users gefragt und vom Client das Passwort übermittelt.

Wichtig für die Sicherheit: Rechner, die die erste Version von Windows 95 als Betriebssystem nutzen und alte NT-Systeme bis einschließlich Service Pack 2 können die Passwörter nur unverschlüsselt übertragen. Wenn Sie derartige Maschinen kommerziell einsetzen, sollten Sie zumindest auf den NT-Rechnern mindestens Service Pack 3 installieren und nach Möglichkeit auf die Win95-PCs ganz verzichten oder NetBSD darauf installieren. Für NT und seine Nachfolger gilt trotzdem: je neuer, desto besser. Aktuell war zur Drucklegung dieses Buchs Servicepack 6a.

user:

Das ist seit der Samba-Version 2.0 die Standardeinstellung. Zuerst einmal muß sich ein Benutzer auf dem Anmelde-Server mit einem gültigen Benutzernamen nebst Passwort authentifizieren. Erst nachdem die eingegebenen Daten als korrekt angesehen wurden, wird die Verbindung zu den angegebenen Diensten und Shares zugelassen. Dabei gilt auch: Ein Gast kann sich in diesem Security-Level nicht mit einem derartigen Server verbinden, weil er eben kein gültiges Benutzerkonto besitzt.

Dieser Parameter kommt auch zum Einsatz, wenn Samba als Domain-Master dienen soll.

server:

Wählt man *server* als Security-Level, leitet der Rechner, auf dem Sie sich anmelden, die Authentifizierung an den örtlichen Primary Domain Controller (PDC) oder einen Backup Domain Controller (BDC) weiter und überträgt ihm diese Aufgabe. Auch hier gilt, daß die Daten über angebotene Shares und Dienste nicht verschickt werden, bevor die Authentifizierung stattgefunden hat. Das wird dann interessant, wenn man die Windows-Dienste auf mehrere Server verteilen will, zum Beispiel, wenn man einen separaten Server für das Filesharing und einen für die Benutzerverwaltung einsetzt. Diese Art

der Verwaltung ist sehr praktisch, wenn man Unix-Rechner in ein Windows-Netz einbindet, auf dem ein bestehender NT-Server die Benutzerverwaltung innehalt.

Ist die Einstellung *security=server* gewählt, muß zudem der Name oder die IP-Adresse des oder der Server mit *password server* = angegeben werden. Ist hier ein falscher Server angegeben, ist das Einloggen nicht möglich:

```
password server = 192.168.10.10, NT-Server
```

domain:

Die Levels *server* und *domain* ähneln sich. Auch hier findet die Authentifizierung am PDC statt, allerdings wird der Verbindungsversuch nur einmal weitergeleitet. Nach der Authentifizierung wird die Verbindung zum PDC wieder freigegeben, um nicht eine der teueren NT-Serverlizenzen ausschließlich zu Authentifizierungszwecken zu verschwenden. Anschließend erledigt der Server, auf den zugegriffen werden soll, die Authentifizierung selbst.

Der Server darf die Verbindung zum PDC wieder freigeben, weil er selbst Mitglied der Domain ist und als solcher die Authentizität der Benutzer sicherstellen kann.

Auch bei *security* = *domain* muß der Name des Passwortservers angegeben werden. Hinter dem Eintrag *server*= muß dabei entweder der Name oder die IP-Adresse der Authentifizierungsmaschine angegeben werden.

9.2.11 Drucken mit Samba

Gesetzt dem Fall, Sie haben nur einen Drucker zu Hause, der an der NetBSD-Maschine angeschlossen ist, können Sie ihn für die Benutzung unter Windows freigeben. Das geht mit:

```
load printers = yes
```

Dann muß Samba noch wissen, welches Unix-Drucksystem man verwendet. Es gibt es zum Beispiel *bsd, cups, hpux, sysv, qnx, aix* und einige andere. Das sieht unter NetBSD naheliegenderweise so aus:

```
printing = bsd
```

Der Abschnitt »printers«

Path:

Hier gibt es einige Besonderheiten. Aus Sicherheitsgründen sollten Sie das Schreiben in dieses Verzeichnis unterbinden. Andererseits ist dieser Pfad gleichzeitig die Druckerwarteschlange, in der der LPD seine Daten bis zu Verarbeitung zwischenlagert.

Setzen Sie dazu bitte zuerst den Pfad. Hier hat der Drucker den Namen *lp0*:

```
path = /var/spool/lpd/lp/lp0
```

Dann verbieten Sie das direkte Schreiben in dieses Verzeichnis:

```
writeable = no
```

Anschließend erlauben Sie den Clients das Anlegen und Löschen von Spool-Dateien, was ja zum Betrieb des LPD notwendig ist. Beispiel:

```
printing = yes
```

Zum Schluß verhindern Sie, daß man den Drucker in der Browse-Liste des Clients sehen kann. Das geht mit:

```
browsing = no
```

9.2.12 Das Browsing in einer Arbeitsgruppe

Um das Browsen über mehrere Subnetze zu ermöglichen, muß jeder dieser Rechner Mitglied der jeweiligen Arbeitsgruppe sein, nicht einer NT-Domain. Zusätzlich muß einer der Samba-Server als Domain Master Browser aufgesetzt werden. Seine Rolle ist es, die Dienste der jeweiligen Arbeitsgruppe allen dazugehörenden Rechnern anzubieten. In einer heterogenen Umgebung muß der Domain Master Browser ein Samba-Server sein. Außerdem darf es nur einen Domain Master je Arbeitsgruppe geben. Um einen Samba-Server als Domain Master zu nutzen, muß dem Samba-Server das in der *global*-Sektion mitgeteilt werden. Der Domain Master sollte zudem der Local Master Browser in seinem lokalen Subnetz sein. Um das zu erreichen, müssen diese Optionen in der »smb.conf« gesetzt sein:

```
domain master = yes
local master = yes
preferred master = yes
os level = 65
```

Wenn Sie es als notwendig ansehen, kann der Domain Master gleichzeitig als WINS-Server arbeiten.

Sie brauchen in jedem Ihrer Subnetze eine Maschine, die als »Local Master« für die Arbeitsgruppe zuständig ist. Das sollte jede NT-Maschine können. Windows 9x ist aufgrund seiner Neigung zu häufigen Systemabstürzen zu diesem Zweck nur sehr bedingt geeignet. Um einen Samba-Server als Local Master einzusetzen, tragen Sie die Parameter wie oben in die Konfigurationsdatei ein. Sie können nur einen Rechner pro

Subnetz als Local Master verwenden. Setzen Sie dafür den Parameter *domain master* auf *no*.

Auch hier ist es unter Windows 2000 anders, besonders, wenn Sie mit der Serverversion arbeiten: Dort gibt es WINS nur noch aus Kompatibilitätsgründen. Die Namensauflösung funktioniert, wie in NetBSD auch, über DNS.

9.2.13 Die zentrale Benutzerverwaltung

Samba kann auch die Benutzerverwaltung, wie in einer »echten« NT-Domain üblich, übernehmen. Das ist allerdings mit etwas Arbeit verbunden.

Wie bereits beschrieben, ist eine NT-Domain nicht aus einem Namens- oder Adreßraum heraus definiert, sondern als logischer Verband von mehreren Clients und Servern, die innerhalb der Domain spezielle Aufgaben im Rahmen der automatischen Netzwerkverwaltung verrichten. Aus diesem Grund gibt es in so einem Netzwerk für einen NT-Server immer etwas zu tun. Dazu gehören zum Beispiel eben auch der Verwaltungs-Overhead des SMB-Protokolls und die ständigen Broadcast-Anfragen der Clients. Das Zentrum einer solchen Domain bildet immer die Ressource: Wenn Sie registrierter Benutzer auf so einer Maschine sind, dürfen Sie diese Ressourcen grundsätzlich auch in Anspruch nehmen.

Um das Verwalten der Benutzer zu zentralisieren, sollten Sie zuerst etwas über die Benutzerverwaltung von Windows NT wissen.

Dort gibt es verschiedene Benutzergruppen, die in lokale und globale Benutzer unterschieden werden.

Globale Gruppen

Das sind verschiedene Benutzer, die alle zu einer einzigen Domain gehören, die mit Hilfe der Gruppe zu einer Verwaltungseinheit zusammengefaßt werden. In dieser Gruppe dürfen sich nur Benutzer aus der gleichen Domain einloggen. Auf der anderen Seite kann man einer globalen Benutzergruppe aber auch Rechte in einer anderen Domain zuweisen.

Lokale Gruppen

In lokalen Gruppen können sowohl normale Benutzer als auch Benutzer aus globalen Gruppen integriert sein. Beide können sowohl aus der gleichen als auch aus einer fremden (trusted) Domain kommen.

Um die Sicherheitsrichtlinien von NT zu erfüllen, hat Microsoft für Windows NT einige Benutzergruppen vordefiniert:

Administratoren	Dieses Gruppe entspricht im wesentlichen der Gruppe *wheel* in NetBSD. Allerdings mit dem Unterschied, daß einzelne Dateien für diese Benutzergruppe gesperrt werden können. Unter Windows NT kann ein solches Konto ablaufen.
Serveroperatoren	Diese Gruppe kann Verwaltungsarbeiten in der Domain ausführen.
Konten-Operatoren	Diese Mitmenschen können Benutzerkonten einrichten und löschen.
Druck-Operatoren	Diese Benutzer dürfen Drucker-Warteschlangen verwalten.
Sicherungs-Operatoren	...dürfen Daten sichern, die freigegeben sind, diese allerdings nach einem Systemcrash nicht wieder auf die Systeme zurückspielen.
Replikatoren	Ein Replikator besitzt das Recht, vorbestimmte Verzeichnisse zwischen NT-Servern und -Workstations zu synchronisieren.
Benutzer	Das sind »Menschen wie Du und ich«, deren einziges Recht einmal ist, sich in der Domain einzuloggen. Sie dürfen Dateiänderungen nur am Arbeitsplatz ausführen, auf dem sie angemeldet sind. Auf dem Server selbst muß so ein Konto zuerst freigeschaltet werden.
Gäste	Besitzen praktisch keine Rechte.
Jeder	Das ist eine Pseudo-Gruppe, zu der alle Mitglieder einer Domain gehören. An diese Gruppe können bestimmte Rechte auf relativ einfache Weise an alle registrierten Benutzer zugewiesen werden.

Wenn Sie einen Samba-Server einrichten, der einen echten NT-Server ersetzen soll, muß er auch mit der Verwaltungsstruktur von Windows NT klarkommen. Das heißt, daß Sie die Benutzerprofile auf Ihrem NetBSD-Server vorhalten müssen, damit der Server die notwendigen Informationen (Bestätigungen und Suchpfade) zur Verfügung stellen kann.

9.2.14 Der Anmeldeprozeß

Wenn sich ein Anwender lokal auf einer Arbeitsstation anmeldet, bekommt er, je nachdem, was in seinem Profilverzeichnis steht, eine individuelle Umgebung angeboten.

Diesen Anmeldeprozeß können Sie in einer NT-Domain zentralisieren. Dazu wird auf dem Server ein »Umgebungsprofil« angelegt, in dem Benutzerprofile, Anmeldeskripten und Stammverzeichnisse der Benutzer verwaltet werden.

In den Benutzerprofilen sind diese Daten enthalten:

Benutzerprofile: Dort steht drin, wie der Desktop des jeweiligen Benutzers aussehen soll. Das geht zentral nur bei Systemen ab Windows NT. Bei 9x-Clients können diese Profile nur lokal gespeichert werden. Also ist auch kein Netzwerkzugriff darauf möglich. Festgelegt werden hier beispielsweise Aussehen und Verhalten des Explorers, die Einträge im Startmenü, was in der Systemsteuerung verstellt werden darf und die Einstellungen für den Drucker. Beim Einsatz von Samba müssen Sie diese Profile manuell auf den Server kopieren, wenn Sie bereits einige Benutzerkonten auf einr NT-Maschine haben.

Anmeldeskripte: Das sind einfache Batchdateien, die ausgeführt werden, wenn der Benutzer sich in der Domain einloggt. Damit werden zum Beispiel freigegebene Netzlaufwerke automatisch mit einem Laufwerksbuchstaben versehen, damit der Benutzer seine Daten immer an derselben Stelle wiederfindet. Diese Skripte funktionieren auch mit Windows-9x-Clients.

Basisverzeichnisse: Das sind die Verzeichnisse, die jedem Benutzer nach der Anmeldung zugeordnet werden. Wenn der Benutzer eine klassische Microsoft-Anwendung wie Word startet, ist das auch das Verzeichnis, auf das beim Speichern oder Öffnen einer Datei zuerst verwiesen wird.

Dieses Verzeichnis kann sich sowohl auf dem lokalen Rechner als auch auf dem Server befinden.

9.2.15 Einrichten eines Domain Controllers

Wenn Sie ein sicheres Netzwerk wollen, sind Sie darauf angewiesen, die Windows-Rechner in einer Domain zusammenzufassen und die Benutzerverwaltung zu zentralisieren. Die im Final Release 1.6 mit NetBSD ausgelieferte Version (2.2.5) ist immerhin in der Lage, einen Domain Controller im Hinblick auf die Domain Logons für NT4/2000/XP-Clients, Windows-9x/ME-Clients im Userlevel, das Verteilen der NT-Systemrichtlinien und das Verwalten von Benutzern und Gruppen für alle Clients, sowohl in NT/2000 als auch für 9x/ME zu ersetzen.

Folgendes kann ein Samba-Domain-Controller nicht:

Als Domain Controller für Windows 2000 kann Samba bisher mangels Kerberos und Active-Directory-Funktionen nicht eingesetzt werden.

Die Datenreplikation mit NT4-Domain-Controllern funktioniert noch nicht.

Es können derzeit noch keine Benutzerkonten mit dem Domain-Usermanager angelegt werden.

Die einzelnen Schritte dazu:

Sie müssen Samba als PDC konfigurieren:

- Die unter Umständen vorhandenen Benutzerprofile auf den Samba-Server kopieren,
- Maschinenkonten einrichten und
- die Clients in die Domain aufnehmen.

Ihre Konfigurationsdatei sollte folgendermaßen aussehen:

```
[global]
netbios name = Servername
workgroup = arbeitsgruppe
os level = 64
preferred master = yes
domain master = yes
local master = yes
security = user
encrypt passwords = yes
domain logons = yes
logon path =
logon drive = H:
logon script = logon.cmd

[netlogon]
path = /usr/pkg/lib/netlogon
read only = no
writelist = ntadmin

[profiles]
path = /home/netlogon/profiles
read only = no
create mask = 0600
directory mask = 0700
```

Benutzerprofile kopieren

Wenn sich ein NT- oder 2000-Workstationrechner im Netz befindet, kopieren Sie die Benutzerprofile aus diesen Maschinen so auf Ihren Samba-Server:

- Um diese Daten transferieren zu können, setzen Sie den Security-Level auf *share*, weil es ja noch keine Benutzerprofile auf dem NetBSD-Server gibt.
- In der »smb.conf« erstellen Sie dann eine Freigabe namens *netlogon*, die les- und beschreibbar sein muß.
- Das dazugehörige Verzeichnis legen Sie unter »/home« an.
- Es gibt in der NT-Systemsteuerung ein Icon namens *System*. Dort klicken Sie auf die Registerkarte *Benutzer* und dort wiederum auf *Kopieren*. Das entsprechende »profiles«-Verzeichnis legen Sie vorher auf dem Server an. In diesem Beispiel ist das *\\Ihr_Server\home\netlogon\profiles*.
- Wenn das passiert ist, ändern Sie die »smb.conf« wieder wie im Beispiel oben.

Das Aufnehmen der Clients in die NT-Domain

Damit man den Samba-Server als Domain Controller verwenden kann, muß Samba wissen, welche Maschinen auf seine Domain zugreifen dürfen. Dies geschieht über die Einrichtung sogenannter Maschinenkonten, die eigentlich Unix-Benutzeraccounts ohne Passwort, Shell und Home-Verzeichnis sind.

In der Windows-Terminologie passiert nichts anderes. Außer daß die Verwaltung bei einem echten NT-PDC in der Registry erfolgt und diese Konten auf einem Samba-Server zweigeteilt sind. Sie brauchen dazu:

Ein Samba-Konto, das sich an derselben Stelle befindet, in sich die NT-Paßwörter befinden. Das ist die Datei »smbpasswd«.

Dann brauchen Sie noch den damit korrespondierenden NetBSD-Benutzeraccount, der in der Datei »/etc/passwd« zu finden ist.

Erstellen können Sie diese Konten auf zwei Arten: Manuell durch Erstellung beider Konten an der Kommandozeile, oder automatisch mit Hilfe eines Maschinenkontos, das Samba automatisch generiert, wenn der Client sich zum ersten Mal mit dem NetBSD-Server verbindet.

Die dazu gehörenden NetBSD-Accounts können entweder automatisch oder manuell erstellt werden.

Zumindest das erste Maschinenkonto sollten Sie manuell erstellen, um zu sehen, was der Rechner macht, wenn ein Windows-Rechner in das Samba-Netzwerk integriert wird. Das geht so:

```
# /usr/sbin/useradd -g 100 -d /dev/null -c "Nickname_der_Maschine" \
                    -s /bin/false [Name Ihres Clients]$
# passwd -l [Name Ihres Clients]$
```

Damit richten Sie einen Benutzeraccount ein, der kein Passwort besitzt, keine Shell und kein Home-Verzeichnis.

Der Nickname kann ein Begriff sein, der etwas über die Funktion der Maschine aussagt. Der Name des Clients ist der NetBIOS-Name des Windows-Clients. Das »$« muß an den Rechnernamen angehängt werden. Samba kann sonst nicht erkennen, daß dieser Account ein Maschinenkonto ist.

Anschließend setzen Sie dieses Kommando ab, mit dem Sie das Konto dem Samba-Server bekanntmachen:

```
# smbpasswd -a -m [Name des Clients]
```

Das automatische Erstellen der Maschinenkonten

Wenn Sie diesen Vorgang automatisieren wollen, geht das mit folgendem Eintrag in die »smb.conf«:

```
add user script= /usr/sbin/useradd -d /dev/null -g 100 -s /bin/false -M %u
```

Dazu brauchen Sie allerdings einen Benutzeraccount mit *root*-Rechten.

Das Einfügen der Clients in die Domain

Wenn das Maschinenkonto manuell erstellt wurde, müssen Sie in der Netzwerkkonfiguration Ihrer Maschine den Domainnamen eintragen, aber die Checkbox *Ein Konto in der Domäne erstellen* nicht anklicken.

Soll das Maschinenkonto automatisch erstellt werden, wählen Sie diese Checkbox an.

Wenn ein Benutzer den Client in der Domain registrieren will, fragt Windows nach einem Konto nebst Passwort, mit dem Sie Mitglied der Domain werden können.

Ein Domain Master für 9x/ME-Clients

Auch mit 9x/ME-Systemen ist das Erstellen einer Domain möglich. Allerdings gibt es ein paar Unterschiede zu einer reinen NT-Domain:

Die Verschlüsselung der Passwörter ist keine Voraussetzung für einen 9x-Logon-Server.

Windows-9x/ME-Clients brauchen außerdem kein Maschinenkonto.

Um das Einloggen von 9x/ME-Clients zu ermöglichen, muß der Parameter *logon home* nach folgendem Schema angewendet werden:

```
logon home =  \\%L\%U
```

Sie können Samba übrigens gleichzeitig als Domain Master für Windows NT/2000/XP und 9x/ME verwenden, ohne daß es Probleme damit gibt, wenn Sie zusätzlich den Parameter *logon home* setzen wie im Beispiel.

9.2.16 Die Konfiguration von Samba mit SWAT

Samba läßt sich auch remote per Webbrowser administrieren. Das Tool dafür heißt SWAT (Samba Web Administration Tool). SWAT wird zusammen mit Samba in einem Stück installiert. Gestartet wird SWAT über den *inetd*, dessen Konfigurationsdatei (»inetd.conf«) mit diesen Einträgen versehen sein muß:

```
swat  stream tcp nowait.400 root /usr/pkg/sbin/swat swat
```

Zusätzlich müssen Sie in »etc/services« folgenden Port mit dieser Eintragung freigeben, um SWAT zu starten:

```
swat 901/tcp
```

Starten Sie den *inetd* neu und verbinden Sie sich mit einem Webbrowser Ihrer Wahl mit dem Samba-Server. Das ist der URL

```
http://Ihr.Samba.Server:901
```

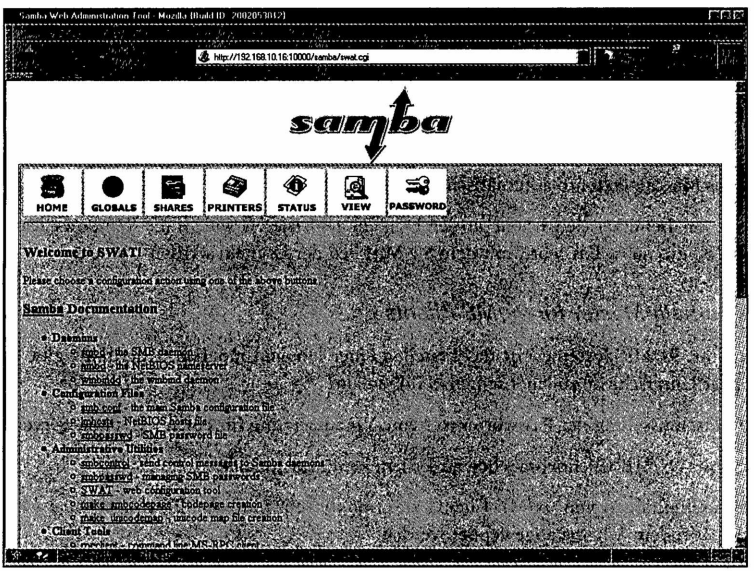


Abb. 9.3: Die Startseite von SWAT

9.2.17 Die grafische Konfiguration

Wenn sie nicht mit SWAT arbeiten wollen, können Sie, einen installierten KDE-Windowmanager vorausgesetzt, Samba auch mit dem Programm *ksamba* verwalten. Es gibt auch noch weitere Samba-Verwaltungsprogramme wie beispielsweise GTK-Samba. Alle haben aber ihre Tücken. Beispielsweise neigt *ksamba* dazu, die Kommentare aus dem »smb.conf« zu entfernen. Man sollte deshalb von diesen Programmen besser Abstand nehmen. Wer unbedingt ein grafisches Frontend verwenden will, sollte besser zum SWAT oder zum Webmin greifen.

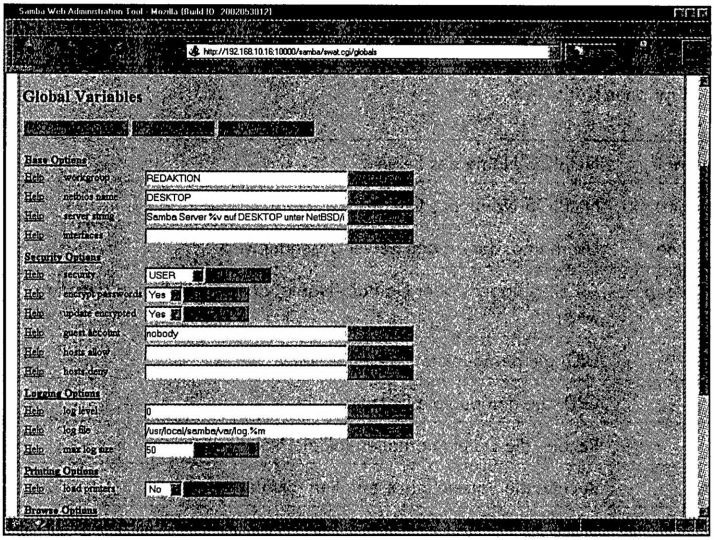


Abb. 9.4: Die Einstellung in SWAT ist komfortabel, hier für die globale Sektion

9.3 Samba-Clients unter NetBSD

von Jörg Braun

9.3.1 Sharity Light

Sharity Light ist seit längerer Zeit in der Version 1.2 erhältlich. Die Entwicklung des Produkts ist abgeschlossen. Das Quellarchiv des Programmpakets ist von *ftp://ftp.obdev.at/pub/Products/Sharity-Light/* erhältlich.

Sharity Light implementiert ein Pseudo-NFS-Interface für Samba-Shares. Da die Kommunikation auf NFS beruht, muß im Kernel deshalb auch NFS eingeschaltet sein, damit das Programm funktioniert. Der eigentliche Vorteil von Sharity Light gegenüber allen anderen Implementationen wie *smbfs* unter Linux oder FreeBSD ist, daß das Programm auf der Anwendungsebene und nicht auf Kernelebene läuft. Das bedeutet, daß, wenn Fehler auftreten, diese nicht zum Absturz des Computers führen. Man kann den Client-Mount immer (im Notfall mit *kill* zwangsweise) beenden und muß ihn nur neu starten.

Nach der Installation aus dem PkgSrc-Verzeichnis »net/sharity-light« sind in »/usr/pkg/sbin« die Programme *shlight* (der Mount-Befehl) und *unshlight* (der *unmount*-Befehl) zu finden. Als Manpage dient eine modifizierte Version von *smbmount*, die in die Sektion 8 abgelegt wird. Wenn man *shlight* mit *root*-Rechten (was schon durch die Position der Binärdatei in »/usr/pkg/sbin« angezeigt wird) ohne Parameter aufruft, wird die Hilfe angezeigt. Der grundlegende Aufruf des Programms erfolgt mit

```
shlight //server/service Mountpoint [Optionen]
```

Mögliche Optionen sind:

-s servername	NetBIOS-Name des Servers.
-c clientname	NetBIOS-Name des Clients.
-U username	Username, der an den Server gesendet wird.
-W workgroup	Arbeitsgruppen- oder Domain-Name, der an den Server gesandt wird.
-u uid	UID, die gemountete Dateien erhalten.
-g gid	GID, die gemountete Dateien erhalten.
-f mode	Rechte, die Dateien erhalten (oktale Notation).
-d mode	Rechte, die Verzeichnisse erhalten (oktale Notation).
-C	Passwort nicht in Großbuchstaben konvertieren.
-P password	Das Passwort *password* verwenden.
-i	Passwort aus der Standardeingabe lesen.
-n	Kein Passwort verwenden. Wenn weder *-P* noch *-n* angegeben ist, wird nach einem Passwort gefragt.
-S	Schreib-/Lese-Zugriff ist sicher. Ohne diese Option werden Dateien geschlossen, wenn ein Wechsel vom Lese- zum Schreibmodus und umgekehrt stattfindet.
-w	Dateien dürfen schreibgeschützt werden.
-e	Die SMB-Befehle *getattrE* und *setattrE* verwenden.
-F .	Den Eintrag ».« im Hauptverzeichnis nicht vortäuschen.
-F ..	Den Eintrag »..« im Hauptverzeichnis nicht vortäuschen.
-h	Anzeigen der Hilfe.
-v	Versionsnummer ausgeben und beenden.
-p port	Zu verwendenden Port angeben (nur für Testzwecke).
-m max_xmit	*max_xmit* angeben (nur für Testzwecke).
-D debug-mode	Debugging aktivieren (Bitmaske in Hex/Dez/Oktal-Notation). Diese Option erzwingt die Operation im Vordergrund.

Ein Aufruf von *shlight* sieht in der Realität so aus:

```
# shlight "//server/joerg@server" /mnt/joerg -U joerg -P joerg
Using port 1042 for NFS.
```

Problematisch mit Zugriffen ist es, wenn die fremden Shares ein Dollarzeichen oder einen Klammeraffen enthalten:

```
# shlight "//joerg/D$joerg" /mnt/joerg -U joerg -P joerg
error connecting to server: [5] Input/output error
```

Man sollte also auf solche Freigaben verzichten, wenn man auf der Unix-Seite Sharity Light einsetzen will.

Zum Testen kann man auch eine lokale Freigabe verwenden. Dann muß man nur anstelle eines Servernamens wie immer im TCP/IP-Netzwerk *localhost* angeben.

Die Sache ist auch heikel, wenn die IP-Adressen der Rechner über DHCP vergeben sind. Arbeitsstationen, die nicht in der Datei »/etc/hosts« eingetragen sind, will Sharity Light nicht finden, auch wenn sie im Samba-Netz vorhanden sind:

```
# shlight "//lappi/c" /mnt/joerg -U joerg -P joerg
lappi: unknown host
```

Der XSM-Browser, der seine Broadcasts über den Samba-Server abwickelt, hat diese Schwierigkeiten nicht.

Das Ausbinden eines Sharity-Mounts aus dem Dateisystem erfolgt mit dem Aufruf von

```
unshlight [-a] | [Mountpunkt]
```

Der Parameter *-a* zeigt an, daß alle mit *shlight* eingebundenen Knoten entfernt werden sollen. Mit der Angabe eines Mountpunkts als Parameter wird nur dieser eine entfernt. Einen Schalter für eine Hilfe oder dergleichen gibt es bei diesem Programm nicht.

9.3.2 xfsamba

Mit der grafischen Oberfläche XFCE wird das Programm *xfsamba* geliefert, das nicht nur unter der XFCE funktioniert, sondern mit jeder beliebigen grafischen Oberfläche eingesetzt werden kann.

xfsamba nimmt als Account für das Login auf anderen Maschinen den aktuellen Namen und das aktuelle Passwort. Das läßt sich aber ändern und somit ist der Zugriff auf Fremdrechner im Windows-Netz mit jedem beliebigen Namen möglich.

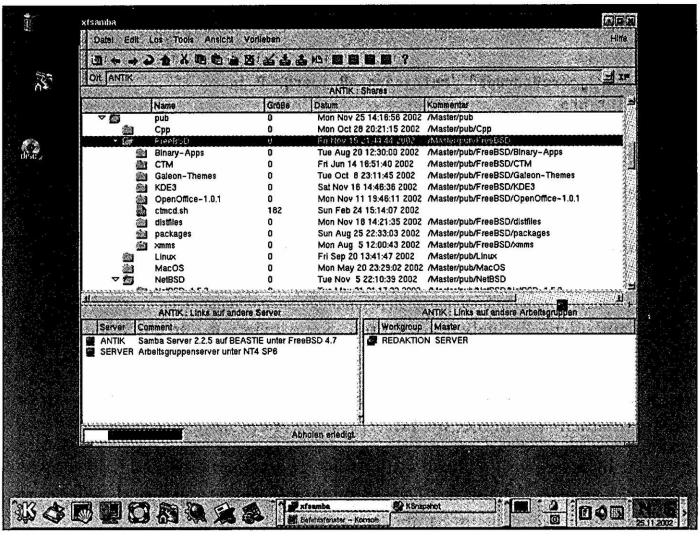


Abb. 9.5: Der grafische Samba-Browser »xfsamba«

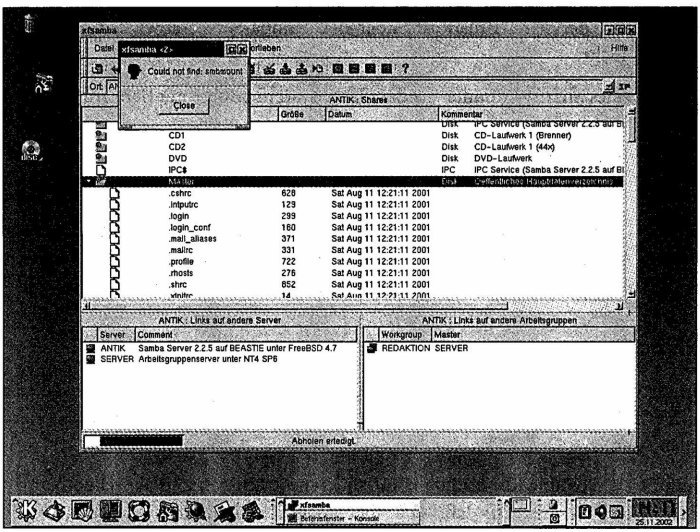


Abb. 9.6: smbmount fehlt unter NetBSD

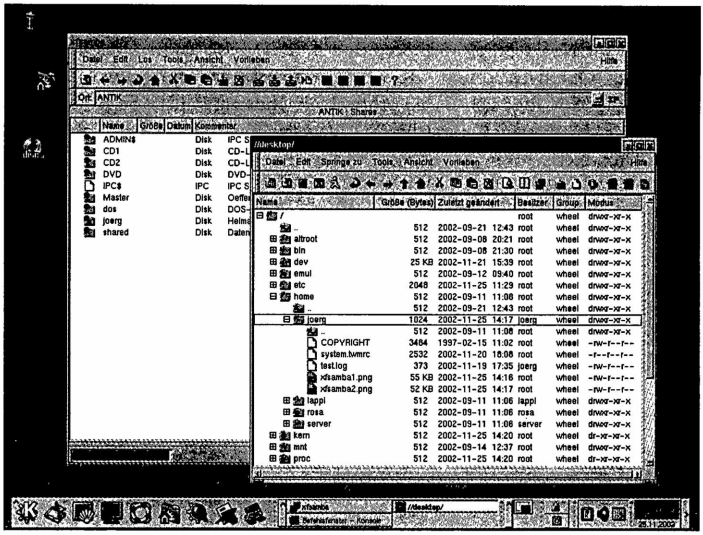


Abb. 9.7: xfsamba mit dem Remote- und dem lokalen Fenster

Im Gegensatz zu Sharity Light hat xfsamba keine Mühe, auch auf Shares, die das »@«-Zeichen enthalten, zuzugreifen. Theoretisch könnte aus dem Programm ein Verzeichnis auch gemountet werden. Allerdings wird dazu das Programm *smbmount* benötigt, das unter NetBSD fehlt. Diese Funktion steht also nur unter Linux und FreeBSD zur Verfügung.

Der Umgang mit dem Programm ist selbsterklärend. Außer einem Remote-Fenster kann parallel auch ein lokaler Verzeichnisbaum eingeblendet werden. Damit lassen sich Kopieraktionen über NetBIOS recht komfortabel durchführen. Solange kein echtes Samba-Client-Programm für Mounts unter NetBSD zur Verfügung besteht, hat man damit die Möglichkeit, Samba unter NetBSD in beiden Richtungen zu betreiben. Schade ist, da man sich zum Verwenden des Programms einen zusätzlichen Window Manager installieren muß, wenn auch die XFCE durchaus nicht schlecht ist.

9.4 MacOS 8/9 im Unix-Netz mit Netatalk

von Michael Steil

Dieses Kapitel widmet sich dem Open-Source-Projekt Netatalk, mit dessen Hilfe man einem MacOS-Client Zugriff auf das Unix-Dateisystem sowie auf Unix-Drucker gewährt.

Wenn man heute von Macintoshs beziehungsweise Rechnern mit MacOS spricht, muß man zwei grundlegend verschiedene Systeme unterscheiden: MacOS bis 9.x und MacOS X (beziehungsweise MacOS X Server). Während MacOS bis zur Version 9.x ein Betriebssystem ist, in dem die 1984 mit der Einführung des Apple Macintosh eingeführten

Konzepte konsequent weitergeführt wurden, ist MacOS X, der offizielle Nachfolger der MacOS-Linie, ein auf NeXT Step/Open Step und 4.4BSD basierendes Unix mit einer MacOS-ähnlichen Oberfläche und Kompatibilitätspaketen für alte MacOS-Anwendungen. MacOS X fügt sich als ein echtes Unix ohne zusätzliche Software in ein Unix-Netzwerk ein. In diesem Kapitel beschäftigen wir uns mit der Integration von MacOS 8/9 ins Unix-Netz.

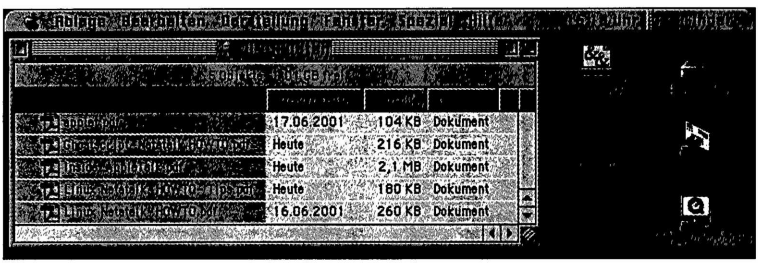


Abb. 9.8: Ein MacOS-Desktop mit einem Netzwerk-Volume und einem Netzwerk-Drucker

Um MacOS 8/9 in ein Unix-Netz zu integrieren, bedarf es bestimmter Zusatzsoftware, da wie auch bei Windows für MacOS eigene Protokolle entwickelt wurden. Bei diesen handelt es sich nicht nur Protokolle wie NFS oder SMB, die auf TCP/IP aufsetzen, Apple hat Anfang der achtziger Jahre sogar ein eigenes in Konkurrenz zu TCP/IP stehendes Protokoll erschaffen, mit dem die ersten Apple Computer (bereits vor dem Macintosh) vernetzt werden konnten.

Wie auch IBM und Microsoft (NetBIOS) hat Apple dann aber in der Zeit der Internetrevolution das proprietäre »AppleTalk« durch TCP/IP ersetzt. Das File-Sharing-Protokoll »AppleShare« wurde aber weiterhin benutzt; es wird nun über TCP/IP getunnelt. Es ist zwar möglich, Unix-Maschinen in ein echtes AppleTalk-Netz zu integrieren, da aber selbst Apple inzwischen zu TCP/IP rät und MacOS X das alte Protokoll nur noch rudimentär unterstützt, soll im folgenden nur darauf eingegangen werden, wie man einen BSD-Rechner so konfiguriert, daß MacOS-Clients in einem TCP/IP-Netz mit Hilfe von AppleShare auf Unix-Ressourcen zugreifen können.

Es gibt momentan drei durchaus unterschiedliche Möglichkeiten, File-Sharing zwischen Mac- und Unix-Computern zu ermöglichen. Es gibt Dave, ein Windows-Programm, das Verzeichnisse über AppleShare freigeben kann. Erhält der Windows-Rechner sein Dateisystem wiederum über das Netz (als SMB-Share), greift der Mac-Rechner im Endeffekt auf das Unix-Dateisystem zu. Allerdings wandern so alle Daten zweimal übers Netz und man benötigt zwingend eine Windows-Umgebung für diese Lösung.

Das Dateikonzept von Apple

Wenn man einen Unix- oder Windows-Anwender fragt, was eine Datei ist, hört man Antworten von »mehrere Bytes hintereinander« bis »eine Abbildung eines eindeutigen Identifizierers auf die Gesamtheit einer Folge von null oder mehr Bit-Oktetts, einer Datums-/Zeitangabe und der Angabe von Zugriffsrechten«. Das stimmt auch alles – für den Dateibegriff von Unix.

Auf Apple-Computern ist eine Datei nicht eine Folge an Byte, sondern zweil Jede Datei besteht aus einem »Resource Fork« und einem »Data Fork«. Im Data Fork (»Datenzweig«) befinden sich die eigentlichen Nutzdaten, und im Resource Fork (»Ressourcenzweig«) diverse zusätzliche Daten wie der Dateityp (MacOS geht nicht nach der Endung einer Datei), der Kommentar, das Etikett, das Icon und so weiter. Der Resource Fork enthält das, was man auf Unix oft »Metadaten« nennt; in KDE und Gnome werden solche Metataden zum Beispiel in zusätzlichen versteckten Daten gespeichert. Wie sieht dieses Dateikonzept nun in der Praxis aus? Nehmen wir eine JPEG-Datei in MacOS. Diese soll den Namen »Hallo« haben – eine Dateinamenserweiterung ist nicht nötig. Der Data Fork hält die Daten, die man in der Unix-Datei erwarten würde. Im Resource Fork befindet sich der Dateityp, ein Code, der aus vier ASCII-Zeichen besteht, in unserem Fall »JPEG«. Des weiteren wird hier auch noch der Code der Anwendung angelegt, die die Datei erstellt hat beziehungsweise mit der die Datei geöffnet werden soll. JPEG-Dateien können natürlich mit mehr als nur einer Anwendung geöffnet werden, deshalb können hier Codes wie »MOSS« (Netscape Communicator) oder »ogle« (QuickTime PictureViewer) stehen.

Abbildung des Apple-Konzepts auf Unix

Haben Sie sich immer gewundert, wieso man als Apple-User im Internet immer ».bin«- oder ».hqx«-Dateien downloaden und diese dann mit *Stufflt* konvertieren muß, anstatt gleich die richtigen Dateien downloaden zu können? Das liegt natürlich daran, daß eine Unix-Datei (Internet-Dateien sind Unix-Dateien) »Single-Fork«-Dateien sind, das heißt, es fehlt der Resource Fork. Eine ».bin«-Datei ist eine Unix-Datei, in der sich beide Forks befinden.

Wie legt nun Netatalk eine Datei im Unix-Dateisystem ab, die eine MacOS-Anwendung schreiben will? Netatalk könnte die Datei im ».bin«-Format ablegen. Dies ist aber sehr ineffizient, da das Anhängen von Informationen an die Datei dann unter Umständen dazu führen kann, daß die vollständige Datei umkopiert werden muß. Sehr viel günstiger ist es, eine zweite Datei anzulegen, in der sich der Resource Fork befindet. Netatalk erstellt in jedem Verzeichnis mit Mac-Dateien ein Unterverzeichnis ».AppleDouble«. Dort werden die Resource Forks in Dateien mit demselben Namen abgelegt. MacOS 8/9 verfolgt übrigens die gleiche Strategie, wenn Mac-Dateien auf FAT-Disketten kopiert werden sollen; das Verzeichnis, in dem die Resource Forks gespeichert werden, heißt hier »resource.frk«. MacOS X macht es ähnlich; das neue Dateisystem UFS unterstützt auch keinen zweiten Fork: Der Resource Fork wird hier in einer versteckten Datei abgelegt, an deren Namen die Zeichen »._« vorangestellt wurden.

Ein weiterer Ansatz ist, wie von Apple vorgeschlagen, einen Rechner mit MacOS X beziehungsweise MacOS X Server ins Netz zu integrieren. Dieser stellt eine ideale Brücke zwischen Unix und Macintosh dar: Setzt man MacOS X als Server ein, kann dieser Verzeichnisse für Unix-Clients über NFS, für Windows-Clients über SMB (Samba) und für MacOS-Clients über AppleShare freigeben, sowie von MacOS 8/9 freigegebene Dateisysteme mounten und diese über NFS verteilen. Andererseits ist es aber nicht möglich, über NFS gemountete Dateisysteme über AppleShare freizugeben. Was Kompatibilität angeht, ist dies sicherlich die beste Lösung. Sie empfiehlt sich vor allem in einem Netz, in dem sich hauptsächlich Apple-Rechner befinden.

Die kostengünstigste und flexibelste Lösung, um Dateien und Drucker freizugeben und auf Apple-Drucker zuzugreifen, ist das freie Paket Netatalk.

Netatalk unterstützt das Freigeben von Unix-Verzeichnissen und Druckern über AppleShare über TCP/IP, sowie, sofern die Unterstützung des Kernels vorhanden ist (zum Beispiel durch das Kernelmodul »appletalk«), das Freigeben in einem echten AppleTalk-Netz. Wie bereits erwähnt, soll dieses Kapitel nur auf ersteres eingehen.

Den Quelltext von Netatalk erhält man auf *http://netatalk.sourceforge.net*. Im Netz werden auch Binärpakete für diverse Betriebssysteme zur Verfügung gestellt. Auf das Kompilieren des Quelltextes und die Installation des Binärpakets will ich an dieser Stelle nicht weiter eingehen. Nach der Installation muß man sich jedoch vergewissern, daß die Datei »/etc/services« die Einträge für die für AppleShare nötigen IP-Ports besitzt. Fügen Sie der Datei also gegebenenfalls folgende Zeilen hinzu:

```
rtmp            1/ddp           # Routing Table Maintenance Protocol
nbp             2/ddp           # Name Binding Protocol
echo            4/ddp           # AppleTalk Echo Protocol
zip             6/ddp           # Zone Information Protocol
afpovertcp 548/tcp              # AFP over TCP
afpovertcp 548/udp
```

Installiert wird Netatalk bei NetBSD wie üblich über das Package, das sich unter »/usr/pkgsrc/net/netatalk« befindet. Nach dem Übersetzen des Ports befindet sich in »/usr/pkg/etc/rc.d« die Startskripten »atalkd«, »afpd«, »timelord« und »papd«, in denen die Netzwerkdienste beim Booten gestartet werden können:

```
#!/bin/sh
#
# PROVIDE: atalkd
# REQUIRE: DAEMON
#
# Appletalk daemons. Make sure not to start atalkd in the background:
# its data structures must have time to stablize before running the
# other processes.
#
```

```
. /etc/rc.subr

name="atalkd"
rcvar=$name
command="/usr/pkg/libexec/netatalk/atalkd"
pidfile="/var/run/${name}.pid"
register="/usr/pkg/bin/nbprgstr"
myhostname=`hostname|sed 's/\..*$//'`
required_files="/usr/pkg/etc/netatalk/atalkd.conf $register"

start_cmd="atalkd_start"

atalkd_start()
{
  if [ -x $command ]; then
    echo 'Starting atalkd.'
    $command $command_args
  fi

  if [ -x $register ]; then
    echo 'Setting AppleTalk info with nbprgstr.'
     $register -p 4 $myhostname:Workstation
     $register -p 4 $myhostname:netatalk
  fi
}

load_rc_config $name
run_rc_command "$1"
```

Listing 9.1: Die Datei »/usr/pkg/etc/rc.d/atalkd«

```
#!/bin/sh
#
# PROVIDE: afpd
# REQUIRE: atalkd
#
# AppleTalk daemons. Make sure not to start atalkd in the background:
# its data structures must have time to stablize before running the
# other processes.
#

. /etc/rc.subr
```

```
name="afpd"
rcvar=$name
command="/usr/pkg/libexec/netatalk/afpd"
etcdir="/usr/pkg/etc/netatalk"
pidfile="/var/run/${name}.pid"
required_files="$etcdir/papd.conf $etcdir/AppleVolumes.default \
$etcdir/AppleVolumes.system"

load_rc_config $name
run_rc_command "$1"
```

Listing 9.2: Die Datei »/usr/pkg/etc/rc.d/afpd«

```
#!/bin/sh
#
# PROVIDE: papd
# REQUIRE: atalkd
#
# AppleTalk daemons. Make sure not to start atalkd in the background:
# its data structures must have time to stablize before running the
# other processes.
#

. /etc/rc.subr

name="papd"
rcvar=$name
command="/usr/pkg/libexec/netatalk/papd"
pidfile="/var/run/${name}.pid"
required_files="/usr/pkg/etc/netatalk/papd.conf"

load_rc_config $name
run_rc_command "$1"
```

Listing 9.3: Die Datei »/usr/pkg/etc/rc.d/papd«

```
#!/bin/sh
#
# PROVIDE: timelord
# REQUIRE: atalkd
#
```

```
# AppleTalk daemons. Make sure not to start atalkd in the background:
# its data structures must have time to stabilize before running the
# other processes.
#

. /etc/rc.subr

name="timelord"
rcvar=$name
command="/usr/pkg/libexec/netatalk/timelord"
pidfile="/var/run/${name}.pid"

load_rc_config $name
run_rc_command "$1"
```

Listing 9.4: Die Datei »/usr/pkg/etc/rc.d/timelord«

Um die Skripten beim Start auszuführen, müssen sie nach »/etc/rc.d« kopiert und in der Datei »/etc/rc.conf« die entsprechenden Startvariablen gesetzt werden:

```
atalkd=YES
papd=YES
afpd=YES
timelord=YES
```

Die Skripten gehen aber davon aus, daß die Konfigurationsdateien sich in »/usr/pkg/etc/netalk« befinden. Darauf ist zu achten!

Der Start der Daemonen kann unter Umständen ziemlich lange dauern. Es gibt aber Möglichkeiten, um den Start zu beschleunigen. Editieren Sie dazu die Datei »/etc/rc.conf«. Wenn Sie den Druckservice nicht verwenden, können Sie hier die Zeile »papd« wie folgt ändern:

```
papd=NO
```

Wenn Sie nur den Dateiservice nutzen wollen, können Sie den Start immens beschleunigen, indem Sie auch den Start von *atalkd* deaktivieren:

```
atalkd=NO
```

Die Suchservices im Netz stehen daraufhin nicht mehr zur Verfügung. Das hat auch den Nachteil, daß in den Programmen »Auswahl« und »Netzwerkbrowser« der Server nicht mehr von selbst erscheint, sondern man die IP-Adresse eingeben muß; die Dateifreigabe funktioniert aber weiterhin ohne Einschränkungen.

Daraufhin sollten die Dämonen *atalkd*, *papd* und *afpd* laufen. Bei einem Portscan (zum Beispiel mit *nmap*) sollte nun Port 548 offen sein:

```
548/tcp    open        afpovertcp
```

Auf dem MacOS-Client muß zunächst einmal das TCP/IP-Netzwerk eingerichtet sein. Im Kontrollfeld »TCP/IP« können Sie entweder einen DHCP-Server eintragen oder die IP-Adresse des Macintoshs fest einstellen. Beachten Sie, daß TCP/IP standardmäßig erst dann aktiviert wird, wenn der Rechner versucht, eine Netzwerkverbindung aufzubauen. Direkt nach dem Start von MacOS wird also ein *ping* von der Unix-Maschine aus fehlschlagen. In den Optionen des Kontrollfelds »TCP/IP« (nur im Benutzermodus mit erweiterten Funktionen sichtbar) können Sie aber den Start von TCP/IP schon beim Booten erzwingen. Für die Funktion des AppleShare-Netzes hat dies allerdings keine Bedeutung.

Wenn Sie nun das Programm *Auswahl* (»Chooser«) aus dem Apple-Menü starten, sollte, nachdem Sie den Punkt *AppleShare* in der linken Liste gewählt haben, der Name des Unix-Servers auf der rechten Seite erscheinen. Sollte dem nicht so sein, überprüfen Sie, ob die AppleTalk-Treiber aktiviert sind (Auswahl rechts unten im Fenster). Für den Fall, daß der Name der Unix-Maschine immer noch nicht erscheint, können Sie auch seine IP-Adresse direkt eingeben. Dies ist zum Beispiel der Fall, wenn Sie nicht alle Dienste von Netatalk gestartet haben. (Anstelle von *Auswahl* können Sie den *Netzwerk Browser* verwenden, um Zugriff auf die Freigaben zu erlangen.)

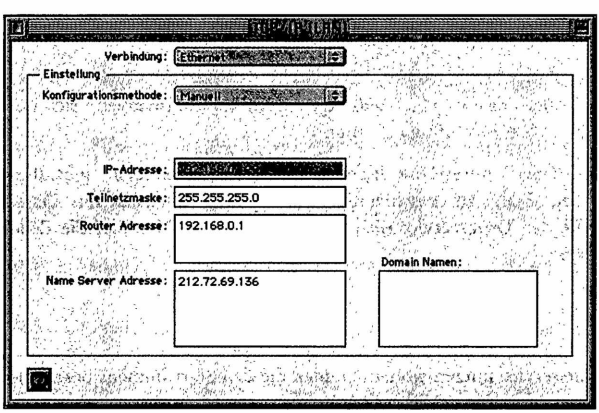


Abb. 9.9: Eine funktionierende TCP/IP-Netzverbindung wird für AppleShare vorausgesetzt

Wählen Sie also den Server aus und klicken Sie auf [OK]. Im darauffolgenden Dialog müssen Sie Benutzernamen und Kennwort eingeben – standardmäßig steht jedem Anwender, der dem Unix-System bekannt ist, sein Home-Verzeichnis als Volume zur Verfügung. Sie können sich auch als Gast anmelden; dann erhalten Sie aber nur Zugriff auf allgemeine Freigaben (standardmäßig keine) und haben nur die Rechte des Unix-

Benutzers *nobody*. Schließlich können Sie noch auswählen, auf welches Volume Sie zugreifen wollen. In der Regel aktiviert man die Checkbox neben dem Home-Verzeichnis, damit das Volume bei jedem Systemstart des Macintoshs automatisch gemountet wird. Nach einem Klick auf [OK] erscheint die Unix-Freigabe auf dem Desktop. Sie können auch mehrere Volumes mounten, dazu müssen Sie die beschriebenen Schritte mehrmals wiederholen.

Das neue Netzlaufwerk verhält sich nun wie eine lokale Festplatte: es können Unix-/Windows-Dokumente darauf geöffnet und Mac-Dokumente und -Anwendungen darauf geschrieben werden. Das Unmounten des Laufwerks erfolgt wie bei entfernbaren Medien mit der Auswurf-Funktion.

Wie bereits erwähnt, ist Netatalk so vorkonfiguriert, daß jeder Anwender Zugriff auf sein Home-Verzeichnis hat. Es ist aber auch möglich, allgemeine Freigaben einzurichten, beispielsweise auf Dateien eines gemeinsamen Projekts. Dafür ist die Konfigurationsdatei »AppleVolumes.default« zuständig, die sich üblicherweise in »/usr/pkg/etc/netatalk« befindet. Sie können der Datei beliebig viele Zeilen hinzufügen, in denen Sie weitere Volumes definieren. Eine solche Definition hat folgenden Aufbau:

```
Pfad Name Optionen
```

Also beispielsweise:

```
/ "UNIX Root" options:ro,noadouble
```

Diese Zeile gibt den kompletten Unix-Baum frei; er kann in MacOS als das Volume »UNIX Root« angesprochen werden. Die Optionen geben an, daß der komplette Baum nur lesbar ist (*ro*), und daß keine unnötigen ».AppleDouble«-Verzeichnisse auf dem Server erstellt werden sollen. Die Option *noadouble* ist übrigens vor allem dann sinnvoll, wenn das Home-Verzeichnis auch von anderen Betriebssystemen aus verwendet werden soll – überflüssige ».AppleDouble«-Verzeichnisse sind unter MacOS nicht sichtbar, sie können aber in einer Unix- oder Windows-Umgebung sehr störend sein. Die Aufstellung aller möglichen Optionen entnehmen Sie bitte den Kommentaren aus der Datei.

Ebenso kann jeder Anwender in seinem Home-Verzeichnis eine Datei ».AppleVolumes« mit dem gleichen Aufbau haben. Ist so eine Datei vorhanden, wird das Home-Verzeichnis nicht mehr als Standard-Volume freigegeben, sondern nur die in der Datei definierten Verzeichnisse:

```
~/macfiles "Server"
```

Diese Zeile bewirkt zum Beispiel, daß für den Benutzer, der diese ».AppleVolumes« in seinem Home-Verzeichnis hat, nicht das komplette Home-Verzeichnis, sondern nur das Unterverzeichnis »macfiles« (unter dem Volume-Namen »Server«) freigegeben wird.

MacOS erkennt die Dateitypen sowie die dazugehörige Anwendung nicht an der Dateinamenserweiterung, sondern an den Informationen im Resource Fork der Datei. Für alle

Dateien im Unix-Dateisystem, die (noch) keinen Resource Fork besitzen, erstellt Netatalk einen, sobald auf die Datei zugegriffen wird. Die Information, welche Dateiendungen in welche Dateitypen umgesetzt werden, steht in der Tabelle »AppleVolumes.system«. Jede Zeile gibt dabei für eine Dateinamenserweiterung den Code für das Dateiformat sowie den Code für die dafür verantwortlich Anwendung an:

```
.doc    "WDBN"  "MSWD"
```

Durch diese Zeile werden zum Beispiel alle Dateien im Unix-Dateisystem, die auf ».doc« enden, als Dateien des Typs »WDBN« mit der Standardanwendung »MSWD« in MacOS sichtbar. Eine entsprechende Tabelle haben auch alle Mac-Browser eingebaut (*Dateihilfsprogramme* im Internet Explorer); für den Fall, daß Sie also Unix-Dateien haben, deren Typ auf dem Mac falsch angezeigt wird, können Sie mit Hilfe der Information aus dem Browser die Datei »AppleVolumes.system« entsprechend editieren.

9.4.1 Drucken unter MacOS

Neben dem Dateisystem will man in einem Netzwerk auch meist Drucker für alle Clients freigeben. Netatalk ermöglicht es, Drucker, die an den Unix-Rechner angeschlossen sind, von MacOS aus sichtbar zu machen.

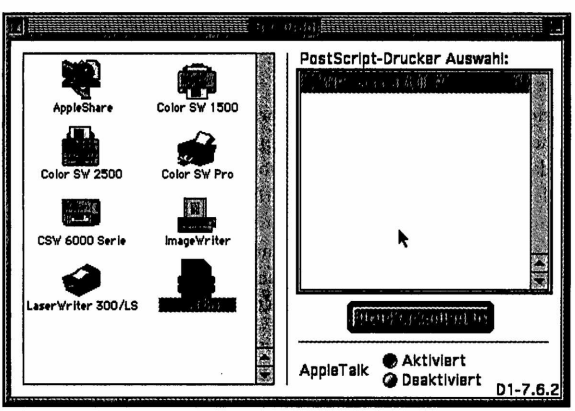


Abb. 9.10: Netzwerkdrucker erreicht man über das Programm »Auswahl«

An dieser Stelle zunächst ein paar Worte über das Druckkonzept von Apple-Computern: Wie auch in Unix läuft in MacOS jegliche Druckerausgabe über das Postscript-Format, das heißt, eine Anwendung oder das Betriebssystem erstellt für jeden Druckjob eine Postscript-Datei, die dann entweder vom Drucker direkt oder vom Druckertreiber interpretiert wird. Dieses Konzept ist nicht weiter verwunderlich: Apple hat schließlich zusammen mit Adobe 1985 das Postscript-Format mit dem Apple LaserWriter eingeführt. Eben dieser LaserWriter wurde im Gegensatz zu den meisten

Druckern aus dem Home-Bereich nicht über einen speziellen Druckeranschluß, sondern über das Netzwerk (AppleTalk) angeschlossen.

Und dies ist genau die Stelle, an der Netatalk in Aktion tritt. Der PAP-Server (Printer Access Protocol) von Netatalk gibt sich als LaserWriter-Netzwerkdrucker aus und nimmt Postscript-Dokumente in Empfang. Diese werden dann je nach Konfiguration an die lokale Druckerwarteschlange gesandt oder direkt an den Drucker geschickt.

Der einfachste Fall ist, wenn bereits ein Drucker konfiguriert und in der Datei »/etc/printcap« eingetragen ist. Netatalk gibt den Drucker dann automatisch als LaserWriter frei und MacOS-Clients können darauf drucken. In solch einem Fall können Sie die nächsten Punkte überspringen.

Manche Druckerserver, wie zum Beispiel CUPS, verwenden die Datei »/etc/printcap« allerdings nicht mehr, sondern haben ihre eigenen Konfigurationsdateien. Wenn Ihr Drucker in Unix bereits funktioniert, können Sie Netatalk durch folgende Zeilen in der Datei »/usr/pkg/etc/netatalk/papd.conf« anweisen, Druckaufträge an den laufenden Druckerserver weiterzuleiten:

```
UNIX Drucker:\
  :pr=lp:\
  :pd=/Pfad/zur/PPD-Datei.ppd:\
  :op=root:
```

Die Option *pr* gibt den Namen des Druckers an und *op* den Unix-Anwender, unter dessen Namen die Aufträge abgewickelt werden sollen. Die bei *pd* angegebene PPD-Datei ist eine Art Druckertreiber für Postscript-Drucker. Sie wird benötigt, um dem MacOS-Client diese Fähigkeiten mitzuteilen. Für Postscript-Drucker erhalten Sie solch eine Datei von Adobe (*ftp://ftp.adobe.com/pub/adobe/printerdrivers/*); das CUPS-Projekt (*www.cups.org*) stellt PPD-Dateien für alle gängigen auch Nicht-Postscript-Drucker bereit. Ist CUPS installiert, befinden sich alle PPD-Dateien bereits im Dateisystem. Achten Sie unbedingt darauf, daß die PPD-Dateien, die ja in reinem ASCII formuliert sind, als Zeilenschaltung das LF-Zeichen (Unix) oder die Kombination CR/LF (DOS/Windows) benutzen und nicht das CR-Zeichen (Mac/Amiga), da sie Netatalk sonst nicht interpretieren kann. PPD-Dateien, die Sie von einer Macintosh-Quelle bezogen haben, sollten Sie zuvor mit

```
# recode mac/cr..l1 Datei.ppd
```

konvertieren.

Sollte der Drucker nun auf diese Art zwar von MacOS ansprechbar sein, aber nicht drucken, überprüfen Sie den Druckernamen. Sie können auch anstelle des Druckernamens wie folgt eine Pipe angeben:

```
:pr=|/usr/bin/lpr -Plp:\
```

Hier wird der Druckerserver nicht über den Socket »/dev/printer«, sondern über das Programm *lpr* angesprochen.

Es ist auch möglich, einen Drucker freizugeben, ohne daß er in Unix eingerichtet ist. Oder wenn Sie den Drucker im Apple-Netz mit anderen Parametern betreiben wollen als im Unix-Netz, können Sie folgende Lösung wählen. Tragen Sie den Drucker wie oben beschrieben in die Datei »papd.conf« ein und geben Sie als Parameter *pr* folgendes ein:

```
:pr=|/bin/cat - >/dev/lp:\
```

Die Postscript-Daten werden dann einfach (ohne Spooling) direkt an den Drucker weitergeleitet. Das funktioniert natürlich nur dann richtig, wenn es sich um einen Postscript-Drucker handelt. Aber auch Nicht-Postscript-Drucker lassen sich so in Betrieb nehmen, indem Sie direkt einen Filter wie GhostScript zwischenschalten:

```
:pr=|/usr/bin/gs -q -sOutputFile=- -sDevice=ljet3 ->/dev/lp:\
```

Dieses Beispiel gibt einen nicht postscriptfähigen HP LaserJet III frei. Für Informationen, wie Sie andere Drucker direkt einbinden können, lesen Sie bitte die Dokumentation zu GhostScript. Ein kleiner Tip noch an dieser Stelle: Sollten Sie einen Postscript-Drucker älteren Datums haben (vor etwa 1992), empfiehlt es sich, diese als Nicht-Postscript-Drucker mit GhostScript-Interpretation freizugeben, da MacOS dazu neigt, zu große und zu komplizierte Postscript-Daten zu erstellen, so daß das Interpretieren durch diese Drucker unter Umständen viel zu lange dauert.

Um den Drucker auf MacOS einzurichten, wird wiederum das Programm »Auswahl« benötigt. Klicken Sie auf »LaserWriter«, wählen Sie Ihren Drucker, der daraufhin auf der rechten Seite erschienen sein sollte, und klicken Sie auf »Drucker anlegen«. Sie werden daraufhin noch aufgefordert, eine PPD-Datei anzugeben, die Sie zum Beispiel von der Unix-Platte nehmen können. Es genügt aber, an dieser Stelle »Allgemein« zu wählen, da der Server bereits die PPD-Datei zur Verfügung stellt. Läuft auf Unix-Seite bereits ein Druckerspooler, kann man das Spooling von MacOS deaktivieren. Von jetzt an können Sie von allen MacOS-Anwendungen aus drucken. Im Druckdialog haben Sie sogar die Möglichkeit, eine Postscript-Datei zu erstellen. Interessanterweise werden in diese Datei im Gegensatz zu den normalen Druckdaten niemals Schriften eingebettet – sollten Sie wie oben beschrieben einen recht alten Postscript-Drucker besitzen, bei dem das Interpretieren der Daten viel zu lange dauert, können Sie versuchen, die Druckausgabe in eine Datei umzulenken, auf das Unix-Dateisystem zu übertragen und dort dann zu drucken.

Dies ist längst nicht alles, was mit dem Paket Netatalk möglich ist. In größeren Netzen ist es zum Beispiel sinnvoll, das Drucken nicht jedem zu erlauben – eine Funktion, die sich in »papd.conf« einstellen läßt. Eine sehr detaillierte Konfiguration an allen Stellen ist über die Dateien in »/etc/atalk« möglich. Wenn Sie darüber mehr erfahren wollen, empfehle ich die Lektüre der vollständigen Dokumentation sowie diverser Internetseiten zu diesem Thema.

9.5 Zugriff auf einen AppleTalk-Drucker

Obwohl Netatalk nicht auf MacOS-Dateifreigaben zugreifen kann, ist es durchaus möglich, von Unix auf einem AppleTalk-Drucker etwas auszudrucken. Die folgende Beschreibung geht von einem funktionierenden TCP/IP- beziehungsweise AppleTalk-Netzwerk aus.

Mit dem Tool *nbplkup* können Sie in Erfahrung bringen, welche Geräte (Computer, Drucker) im Netz vorhanden sind. Jede Zeile der Ausgabe hat folgendes Format:

```
Name:Typ Adresse
```

Sie können die Ausgabe auch nach Netzwerkdruckern filtern, indem sie *nbplkup* wie folgt aufrufen:

```
nbplkup :LaserWriter
```

Wenn Sie also nun den Namen des Druckers wissen, erledigen Sie das Drucken einer Postscript-Datei wie folgt:

```
# pap -p "Druckername" testseite.ps
```

Der Druckername kann weggelassen werden, wenn sich im Home-Verzeichnis des aktuellen Anwenders eine Datei mit dem Namen ».paprc« befindet, die den Druckernamen enthält. Ohne Angabe einer Datei liest *pap* von der Standardeingabe.

Der Status eines Netzwerkdruckers kann über *papstatus* abgefragt werden:

```
# papstatus -p "Druckername"
```

Ein *ping* erfolgt über *aecho*:

```
# aecho "Druckername"
```

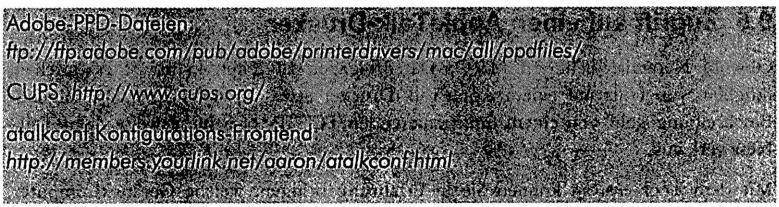
Links

Netatalk Homepage: *http://netatalk.sourceforge.net/*

Faq-o-matic: *http://www.zettabyte.net/fom-serve/netatalk/cache/1.html*

ApplePrint – How to setup the printer using AppleTalk: *http://www.giub.unibe.ch/~eugster/appleprint.html*

Ghostscript/Netatalk HOWTO: *http://ohm.apl.washington.edu/~dunlap/misc/printing/linux/ghostscript.html*



9.6 AFS

von Dr. Lex Wennmacher

Ähnlich wie bei NFS handelt es sich bei AFS ebenfalls um ein verteiltes Dateisystem, bei dem auf Dateien via Netzwerk zugegriffen wird. Das *Andrew File System* (AFS) wurde am MIT entwickelt, später von Transarc vermarktet und ist heute als *OpenAFS* im Quelltext verfügbar. Allerdings hat es seinen Weg noch nicht in den NetBSD-Kernel gefunden, da es den dort geforderten Qualitätskriterien nicht entspricht.

AFS verfügt aber über eine Reihe von Erweiterungen im Vergleich zu NFS:

- Alle Dienste sind kerberosbasiert, was einen sicheren Datenaustausch gewährleistet.
- Dezentrale Administration.
- Lokales Zwischenspeichern.
- Access-Control-Listen (ACL), sie bieten wesentlich feinkörnigere Möglichkeiten Zugriffsrechte zu setzen als Standard-Unix Owner/Group/World Bits).
- Speicherplatz ist unabhängig vom Namensraum.

Zwar ist OpenAFS nicht nach NetBSD portiert (und wird es wohl auch nicht werden), aber es gibt *Arla*, einen bewährten AFS-Clone im Paketsystem (unter »pkgsrc/net/arla«). Der AFS-Client von Arla (bestehend aus einem ladbaren Kernelmodul und dem *arlad*-Daemon) funktioniert zuverlässig, erreicht aber nicht ganz die Leistung von OpenAFS, da zum Beispiel das Cachen auf Dateiebene vorgenommen wird.

Das Arla-Package ist übrigens eines der wenigen Software-Pakete, das die Kernel-Quellen benötigt und einen festen Pfad zu ihnen gesetzt haben möchte. Dabei wird aber nicht der reguläre Pfad »/usr/src/sys«, sondern der Systemlink »/sys« überprüft, der auf dieses Verzeichnis weist.

Sind die Quellen nicht oder an falscher Stelle installiert, meldet das Source-Package:

```
==> arla-0.35.9 requires kernel sources available under $BSDSRCDIR/sys (or
/sys)
```

und beendet sich. Sie sollten bei Bedarf also die angegebene Umgebungsvariable setzen.

10 Die Shell

von Alexander Mayer

Für die einen ist die Shell das mächtigste Werkzeug von Unix, für die anderen ein kryptisches Programm, hinter dem das Ziel steckt, Bequemlichkeit zu eliminieren. Die Kritik, die Shell habe eine hohe Einstiegshürde, ist nur berechtigt, wenn man sich zum Ziel setzt, den vollen Funktionsumfang auszunutzen. Doch muß man ja nicht gleich Profi werden. Bereits auf wenigen Buchseiten kann man einen guten Einblick in die Mächtigkeit der Shell bekommen. Dieses Kapitel führt deshalb in die wichtigsten Grundfunktionen der Shells auf NetBSD ein und zeigt, wie man sich die Shell so bequem einrichtet, daß man sich wie zu Hause fühlen kann.

10.1 Überblick über die Shells

Shell ist nicht gleich Shell. In der langen Geschichte von Unix wurden immer wieder neue Shells und verbesserte Versionen entwickelt, so daß man heute von mindestens fünf weit verbreiteten Shells sprechen kann. Hinzu kommen jeweils verschiedene Versionen, die teilweise sogar nicht einmal abwärtskompatibel zu den Vorgängern sind. Rechnet man nun auch noch die Clones dazu, summieren sich die verschiedenen Shells auf eine unüberschaubare Zahl. Es herrschen also babylonische Verhältnisse, die allerdings vor allem den Shellprogrammierer tangieren. Reine Anwender, die nur mit ihrer Login-Shell zurechtkommen müssen, werden auf keinerlei Probleme stoßen.

Angesichts dieser Verwirrung stellt sich automatisch die Gretchenfrage: Welche Shell ist denn die beste? Natürlich kann man auf diese Frage keine zufriedenstellende Antwort geben. Entscheidend sind jeweils die Präferenzen des einzelnen. Werfen wir also einen Blick auf die verschiedenen Shells in historischer Reihenfolge.

Bourne-Shell

Die Bourne-Shell, benannt nach ihrem Erfinder Steve Bourne, ist die erste Shell von Unix. Sie war damals aufgrund ihrer mächtigen Programmiersprache eine Sensation und hatte sicherlich einen erheblichen Anteil am Erfolg dieses Betriebssystems. Für heutige Verhältnisse reicht allerdings ihr Funktionsumfang, vor allem bei den interaktiven Fähigkeiten, kaum aus. Beispielsweise stehen an der Kommandozeile nur die Tasten [Backspace] und [Strg]+[U] zur Verfügung – Cursortasten werden nicht benutzt. Als Login-Shell sollte die Bourne-Shell also nur in Notfällen fungieren. In der Shellprogrammierung dagegen wird sie sehr häufig eingesetzt.

C-Shell

Die klassische Login-Shell von BSD-Systemen ist seit ihrer Einführung in 3BSD die C-Shell. Sie versucht, sich bei der Programmierung an die Syntax von C anzulehnen. Wegweisend waren vor allem die stark ausgebauten interaktiven Fähigkeiten der Kommandozeile: Funktionalitäten wie der History-Mechanismus, die Job-Verwaltung und die Dateinamenvervollständigung gelten heute als Standard. NetBSD liegt die Original-Implementierung bei, die bis heute keine wesentlichen Erweiterungen erfuhr.

Da die C-Shell in NetBSD nicht nur die Login-Shell des Benutzers *root*, sondern auch die Standard-Shell für alle anderen Benutzer ist, wird sie ausführlich in diesem Kapitel behandelt. Allerdings sei hier nicht verschwiegen, daß andere Shells im Funktionsumfang deutlich überlegen sind – Befürworter loben hingegen das klare Design und die Überschaubarkeit. Empfehlenswert ist die C-Shell vor allem für den Benutzer *root*, da sie mittlerweile als ausgereift und somit stabil gelten kann und darüber hinaus wenig unfreiwillige »Fallen« enthält, die eine unkontrollierte Aktion auslösen könnten. Beispielsweise sei hier das Stichwort Dateinamenvervollständigung genannt.

Korn-Shell

Während sich die C-Shell von den wichtigsten Prinzipen der Bourne-Shell getrennt hat, ist die Korn-Shell wieder fast vollständig abwärtskompatibel. Wagt man einen Blick in den Quelltext, stellt man sogar fest, daß der Entwickler und Namensgeber David Korn den Bourne-Shell-Quellcode bei AT&T einfach weiterentwickelt hat:

```
sh by S. R. Bourne, rewritten by David Korn
```

Die Vorteile der Korn-Shell liegen vor allem in der verbesserten Programmierbarkeit und der aufgewerteten Kommandozeile. Auf den meisten kommerziellen Unix-Systemen ist die Korn-Shell die Login-Shell. Auch Programmierer verwenden vorzugsweise die Korn-Shell. In NetBSD ist der OpenSource-Clone Public Domain Korn-Shell (*pdksh*) enthalten, der gegenüber dem Vorbild von David Korn besonders hinsichtlich der Konfigurierbarkeit punktet. Allerdings ist die *pdksh* (noch) nicht vollständig kompatibel.

Bourne-Again Shell

Auf GNU/Linux-Systemen hat sich die Bash als Standard bewährt. Sie versteht sich, wie der Name Bourne-Again-Shell bereits andeutet, als konsequente Weiterentwicklung der Bourne-Shell, hat sich jedoch von dieser mittlerweile so weit entfernt, daß man eigentlich von einer Shell sprechen müßte, die viele Elemente der C-Shell und vor allem der Korn-Shell integriert. Besonders sticht bei der Bash der bequeme und flexibel konfigurierbare Zeileneditor hervor.

Z Shell

Die Z Shell ist die »eierlegende Wollmilchsau« unter den Shells: Dieses von Paul Falstad initiierte OpenSource-Projekt hat sich zum Ziel gesetzt, die Bourne-, die C- und die Korn-Shell vollständig emulieren zu können. Darüber hinaus enthält sie eine Fülle von Erweiterungen und wesentliche konzeptionelle Verbesserungen, die dem Programmierer das Herz aufgehen lassen. Leider ist in diesem Buch kein Platz für eine ausführliche Einführung.

Dieses Kapitel legt den Fokus auf die Bourne-Shell, die C-Shell, die Korn-Shell und die Bash. Dabei werden wir uns hauptsächlich mit der Bedienung der Kommandozeile beschäftigen. Beachten Sie bitte, daß hier keine Rücksicht auf ältere Versionen genom-

men werden kann. Wir beschränken uns auf den Lieferumfang der zur Drucklegung dieses Buches aktuellen Versionen der wichtigesten Shells bei NetBSD 1.6:

Shell	Version	Pfad
Bourne-Shell	–	/bin/sh
C-Shell	–	/bin/csh
Korn-Shell	Public-Domain-Korn-Shell 5.2.14	/bin/ksh
Bourne Again Shell	Bash 2.05	/usr/pkg/bin/bash

Die nächsten beiden Kapitel bilden Grundlagen, die weitgehend in allen Shells gelten. Danach gehen wir auf die einzelnen Shells genauer ein. Beachten Sie, daß die Kapitel meist für mehrere Shells geschrieben sind, um Redundanz zu vermeiden. Es geht aber klar aus dem Text hervor, welche Shell welche Features unterstützt.

Die Login Shell

Der Begriff »Login-Shell« definiert die Shell, die beim Einloggen verwendet wird. Durch den Aufruf mit dem Namen kann jede Shell wie andere Programme auch ausgeführt werden. Welche Shell beim Login automatisch aktiviert ist, legt man mit dem Programm *chsh* fest.

Eine Login Shell muß nicht nur auf dem System installiert sein. Sie muß inklusive des Pfads auch in der Datei »/etc/shells« eingetragen sein.

10.2 Grundlagen

Bevor wir zu den einzelnen Shells kommen, hier einige Grundlagen, die für die Arbeit mit allen Shells elementar sind. Sie können also die in diesem Kapitel präsentierten Beispiele bis auf wenige Ausnahmen in jeder Shell nachvollziehen.

10.2.1 Kommandos

Der berühmte blinkende Cursor der Shell fordert den Benutzer auf, ein Kommando einzugeben. Kommen wir diesem dringenden Wunsch nach und geben etwas ein:

```
$ cd /bin
```

Damit wechselt man in das Verzeichnis »/bin«, das grundlegende Unixtools enthält. Betrachten wir nun die Zeile aus der Perspektive der Shell: Es handelt sich dabei um zwei von einem Leerzeichen getrennte Wörter, wobei das erste ein Befehl und das zweite ein Parameter ist. Ein Befehl und die dazugehörigen Parameter müssen immer mit

mindestens einem Trennzeichen getrennt werden, damit die Shell ein Unterscheidungsmerkmal besitzt. Welche Zeichen als Worttrenner fungieren, wird in der Variable *IFS* festgelegt, üblich sind das Leerzeichen oder der Tabulator (siehe nächstes Kapitel).

Bei der Eingabe eines Kommandos ist es immer wichtig, im Hinterkopf zu behalten, wie die Shell die eingegebene Zeile bearbeitet. Nehmen wir an, wir übergeben der Shell folgendes Kommando:

```
$ echo                Hallo Welt!
Hallo Welt!
```

echo ist ein Shellbefehl, der Text auf den Bildschirm ausgibt. Wohin sind die führenden Leerzeichen verschwunden? Rufen wir uns noch einmal in Erinnerung, wie die Shell vorgeht: Sie spaltet das Kommando mit Hilfe von Trennzeichen in Wörter auf. Dabei ist es unerheblich, wie viele Trennzeichen sich zwischen zwei Wörtern befinden. Die Leerzeichen werden also ignoriert. Doch woher kommt dann das eine verbliebene Leerzeichen? Dies erzeugt der Befehl *echo*, da er alle übergebenen Parameter durch genau ein Leerzeichen getrennt ausgibt.

Möchte man mehrere Leerzeichen mit *echo* ausgeben, muß man doppelte oder einfache Anführungszeichen (siehe Kapitel 10.2.7) verwenden:

```
$ echo "               Hallo Welt!"
              Hallo Welt!
```

Beachten Sie, daß hier *echo* nur genau einen Parameter erhält. Auch wenn es sich bei »Hallo Welt!« um zwei Wörter im grammatikalischen Sinn handelt, spricht man in der Welt der Shell hier von einem *Wort*.

10.2.2 Ein- und Ausloggen

Nach getaner Arbeit möchte man sich natürlich wieder ausloggen. Dies geschieht am einfachsten mit der Tastenkombination [Strg]+[D], gedrückt in einer leeren Zeile. Je nach Shell funktioniert dies aber nicht immer. Beispielsweise ist die C-Shell in NetBSD so konfiguriert, daß sie folgende Meldung ausgibt:

```
$ ^D
Use "logout" to logout.
```

Man versucht damit, ein unbeabsichtigtes Ausloggen zu verhindern, da diese Tastenkombination noch mit einer anderen Funktion belegt ist. Garantiert immer funktioniert die Eingabe von *exit*, gefolgt von [Enter]:

```
$ exit
```

10.2.3 Variablen

Eine Variable liefert dem Benutzer die Möglichkeit, einen Wert für die Lebensdauer der Shell zu speichern. Variablen im allgemeinen sind deshalb für den Shellprogrammierer elementar. Doch auch der Anwender kommt mit ihnen in Form von Umgebungsvariablen in Berührung. Dies sind Variablen, die entweder wichtige Informationen für Programme enthalten oder der Konfiguration der Shell dienen. Eine der bekanntesten Umgebungsvariablen ist *HOME*. Sie enthält das Homeverzeichnis des aktuellen Benutzers. Den Inhalt einer Umgebungsvariable kann man durch Voranstellen des Dollar-Zeichens erreichen:

```
$ echo $HOME
/home/quark
```

Natürlich kann man einer Umgebungsvariable auch einen neuen Wert zuweisen. Dies geschieht in der C-Shell mit dem Aufruf

```
setenv <Umgebungsvariable> <Wert>
```

und in allen anderen Shells mit

```
export <Umgebungsvariable>=<Wert>
```

Jede Shell besitzt eine unterschiedlich große Anzahl von Umgebungsvariablen, die bereits beim Start definiert werden. Beispielsweise gibt *MAIL* an, in welcher Datei sich das lokale E-Mail-Postfach des aktuellen Benutzers befindet:

```
$ setenv MAIL $HOME/mbox
```

beziehungsweise

```
$ export MAIL=$HOME/mbox
```

Einen Überblick über alle derzeit geltenden Umgebungsvariablen liefern *setenv* und *export* ohne Parameter.

Mit Shell-Variablen (»normalen Variablen«) werden wir im folgenden nur bei der Konfiguration in Berührung kommen, deshalb soll hier noch kurz deren Syntax für die Wertzuweisung geklärt werden. In der C-Shell muß man

```
set <Variable>=<Wert>
```

und in allen anderen Shells

```
<Variable>=<Wert>
```

eingeben. Betrachten Sie dazu einige Beispiele:

```
sh_ksh_bash$ variable=wert
sh_ksh_bash$ echo $variable
wert
csh$ set variable=wert
csh$ echo $variable
wert
```

Vielleicht fragen Sie sich, was der Unterschied zwischen einfachen Variablen und Umgebungsvariablen ist. Eine Umgebungsvariable wird von der Shell bei einem Programmstart in dessen sogenannte Umgebung kopiert, so daß das Programm auch darauf zugreifen kann. Bei einfachen Variablen geschieht dies nicht. Natürlich müssen nicht alle Variablen, die der Konfiguration der Shell dienen, umbedingt in die Umgebung eines Programms kopiert werden. Es hat sich jedoch eingebürgert, dies trotzdem zu tun.

10.2.4 Kommandoketten

Die Shell arbeitet zeilenorientiert – anders ausgedrückt heißt dies, daß alle Wörter nach dem Befehl als dessen Parameter angesehen werden. Um trotzdem mehrere Kommandos in einer Zeile unterzubringen, setzt man das Semikolon:

```
<Kommando1> ; <Kommando2> ; <Kommando3> ; [...]
```

Folgendes Beispiel erzeugt ein neues Verzeichnis und verschiebt dann alle Dateien im aktuellen Verzeichnis in das neue:

```
$ mkdir /alte_version; mv * /alte_version
```

Angenommen, wir besitzen keine Schreibrechte, um das neue Verzeichnis zu erstellen. In diesem Fall versucht *mv* unnötigerweise, die Dateien zu verschieben und beschwert sich mit einer Fehlermeldung. Natürlich entsteht durch diesen Fehler hier kein Schaden, wir haben es lediglich mit einem Schönheitsfehler zu tun. Allerdings gibt es Fälle, in denen solche Probleme durchaus schmerzhafte Folgen haben können. Denken Sie nur beispielsweise an folgende Situation:

```
$ cdrecord dev=/dev/cd0d -data netbsd.iso -fix; rm netbsd.iso
```

Wir brennen hier eine Datei auf CD und löschen die Datei danach. Falls der Brennvorgang nicht vollständig beendet werden konnte, löscht die Shell trotzdem die Datei! In solchen Fällen wäre es vorteilhaft, man könne der Shell beibringen, das zweite Kommando nur dann auszuführen, wenn das erste erfolgreich war. Glücklicherweise gibt jedes Unix-Programm und jeder Shellbefehl einen sogenannten Exitcode zurück. Hat dieser den Wert 0, wurde das Kommando normal beendet, jeder andere Wert deutet

auf einen Fehler hin. Um den Ursachen auf den Grund zu gehen, unterscheiden hier die meisten Programme zwischen verschiedenen Fehlercodes. Ein Blick in die Manpage dürfte genügen. Der Wert des Exitcodes des zuletzt ausgeführten Befehls kann über die Variable *$?* ausgelesen werden:

```
$ echo $?
0
```

Die Shell bietet uns nun mit »&&« eine Möglichkeit, den Exitcode derart auszunutzen, daß das zweite Kommando nur bei Erfolg des ersten ausgeführt wird:

```
<Kommando1> && <Kommando2> && <Kommando3> && [...]
```

Unser Beispiel müßte also folgendermaßen aussehen:

```
$ cdrecord dev=/dev/cd0d -data netbsd.iso -fix && rm netbsd.iso
```

Analog sorgt »||« dafür, daß das zweite Kommando nur dann ausgeführt wird, wenn das erste einen Exitcode ungleich 0 zurücklieferte:

```
<Kommando1> || <Kommando2> || <Kommando3> || [...]
```

Hilfreich ist dieses Konstrukt vor allem in Shellskripten, da sich damit sehr kompakt Fehlermeldungen realisieren lassen:

```
$ cp buch.txt /data/cul || echo Kein Schreibzugriff auf das Verzeichnis.
```

10.2.5 Befehle zusammenfassen

Es gibt in der Shell zwei verschiedene Methoden zum Zusammenfassen mehrerer Befehle:

```
( <Kommando1>; <Kommando2>; [...] )
{ <Kommando1>; <Kommando2>; [...] ; }
```

Beide Konstrukte erledigen dieselbe Aufgabe, unterscheiden sich aber in einigen Details: Die runden Klammern (...) starten die angegebenen Kommandos in einem eigenen Prozeß, die geschweiften Klammern { ... ; } in der aktuellen Shell. Man bezeichnet den Prozeß, den die Shell bei (...) erzeugt, als Subshell. Zu beachten ist außerdem die Syntax: Bei den geschweiften Klammern folgt nach der öffnenden Klammer zwingend ein Leerzeichen, zusätzlich muß vor der schließenden Klammer ein Semikolon stehen. Für die runden Klammern gibt es keine Sonderregelungen.

Warum besitzen die geschweiften Klammern solche seltsam anmutenden Syntaxregeln? Dies kommt von der Tatsache, daß sie anders als die runden Klammern keine Metazeichen der Shell sind. Sie sind vielmehr ein Schlüsselwort, also so etwas wie ein Shellbefehl. Dieser Unterschied ist historisch begründet: Sehr alte Bourne-Shells kannten { ... } nicht. Nun stellen Sie sich vielleicht die Frage, warum Subshell oder nicht Subshell überhaupt eine Rolle spielt. Von Vorteil ist eine Subshell immer, wenn Kommandos in einer »geschützten« Umgebung gestartet werden sollen. So wirken sich beispielsweise Verzeichniswechsel innerhalb einer Subshell nicht auf die aufrufende Shell aus. Betrachten wir dazu folgendes Beispiel:

```
$ pwd
/mnt/home/quark
$ cd texte
$ ls x*
x   xx
$ cd ..
```

Durch die runden Klammern können Sie sich hier die letzte Zeile sparen:

```
$ pwd
/mnt/home/quark
$ (cd texte; ls x*)
x   xx
$ pwd
/mnt/home/quark
```

Mit den geschweiften Klammern funktioniert dieses Beispiel jedoch nicht:

```
$ pwd
/mnt/home/quark
$ { cd texte; ls x*; }
x   xx
$ pwd
/mnt/home/quark/texte
```

Besonders in Verbindung mit »&&« oder »||« machen die Klammern Sinn:

```
$ { make && ./programm; } || vi programm.c
```

Eine solche Zeile erledigt für den C-Programmierer die Kompilierung und anschließende Ausführung des Programms. Erzeugte der Compile-Vorgang mit *make* einen Fehler, wird das Programm nicht ausgeführt und stattdessen der Quelltext mit vi editiert. An diesem Beispiel können Sie auch erkennen, daß die Klammern den Exitcode des zuletzt in der Klammer aufgerufenen Programms zurückliefern.

10.2.6 Externe Programme

Neben Shellbefehlen wie *cd*, die von der Shell direkt ausgewertet werden, ist es selbstverständlich auch möglich, beliebige Programme auszuführen. Beispielsweise möchten wir die Manpage der Korn-Shell betrachten:

```
$ man ksh
```

Hier startet die Shell das Programm *man* und übergibt ihm den Parameter *ksh*. Angesichts der Tatsache, daß NetBSD ausführbare Programme an verschiedene Orte im Dateisystem verteilt (beispielsweise »/bin«, »/usr/bin« oder »/usr/pkg/bin«), stellt man sich natürlich die Frage, woher die Shell weiß, in welchem Verzeichnis sich das Programm befindet. Einen Hinweis darauf liefert uns folgender Aufruf:

```
$ echo $PATH
/home/quark/bin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin:/usr/pkg/bin:/usr
/pkg/sbin:/usr/games:/usr/local/bin:/usr/local/sbin
```

Die Variable *PATH* enthält alle Verzeichnisse, in denen nach Programmen gesucht werden soll. Die einzelnen Werte werden dabei durch einen Doppelpunkt getrennt.

Beachten Sie, daß die Shell nicht wie in DOS oder Windows üblich, automatisch im aktuellen Verzeichnis sucht. Wünschen Sie dies, so muß sich ».«, der symbolische Name für das aktuelle Verzeichnis, in *PATH* befinden. Aus Sicherheitsgründen sind die Shells von NetBSD jedoch standardmäßig nicht so konfiguriert. Um die Problematik dahinter zu verstehen, erzeugen wir ein kleines Testszenario. Als erstes kopieren wir das Programm */bin/rm* in das Homeverzeichnis und nennen es »mv«:

```
$ cd
$ cp /bin/rm mv
$ ls mv
mv
```

Im zweiten Schritt erweitern wir *PATH* um das aktuelle Verzeichnis. Dazu rufen wir in der C-Shell

```
$ setenv PATH .:$PATH
```

und in allen anderen Shells

```
$ export PATH=.:$PATH
```

auf.

Um keine wichtigen Daten zu zerstören, erzeugen wir eine Testdatei mit *touch*. Dieses Unixtool erzeugt leere Dateien, falls sie noch nicht angelegt sind:

```
$ touch xxx
```

Möchten wir nun diese Datei umbenennen, erleben wir eine Überraschung:

```
$ mv xxx yyyy
mv: yyy: No such file or directory
$ ls xxx
ls: xxx: No such file or directory
```

Die Datei »xxx« ist offensichtlich verschwunden. Betrachten wir zur Erläuterung die Vorgehensweise der Shell in einer vereinfachten Form:

- Die Shell analysiert die Zeile und findet den Befehl *mv*. Da dies kein Shellbefehl ist, beginnt die Suche in *PATH*.
- Da in der in *PATH* angegebenen Reihenfolge gesucht wird, wird die Shell bereits im aktuellen Verzeichnis bei der Datei *mv* fündig – unser vormals kopiertes »/bin/rm«.
- *rm* wird aufgerufen, woraufhin dieses Programm die Datei »xxx« löscht und keine Datei »yyy« findet.

Nun wird Ihnen sicherlich klar sein, warum es sinnvoll ist, das aktuelle Verzeichnis nicht in *PATH* aufzunehmen. Dies gilt natürlich insbesondere für den Benutzer *root*. Übrigens können Sie Programme im aktuellen Verzeichnis folgendermaßen trotzdem aufrufen:

```
./programm
```

10.2.7 Quoting

Die Shell nimmt viele Zeichen im Zeichensatz in Anspruch. Beispielsweise kennen wir bereits die Bedeutung von »*«, »$« oder »?«. Was müssen wir jedoch tun, um diese Zeichen ohne Interpretation der Shell zu verwenden? Hierfür gibt es das sogenannte Quoting.

Am einfachsten ist das Quoting, wenn nur ein paar einzelne Zeichen verdeckt werden sollen. Hierfür muß man dem gewünschten Zeichen lediglich einen Rückstrich (engl. backslash) voranstellen:

```
$ echo Das kostet Dich mindestens \$10,23.
Das kostet Dich mindestens $10,23.
```

Spätestens wenn mehrere Zeichen in einer Zeile gequotet werden müssen, wird dieses Verfahren jedoch sehr umständlich. Deshalb gibt es das vollständige Quoting, das mehrere Zeichen mit Hilfe von einfachen Anführungszeichen einschließt:

```
' <Zeichenkette> '
```

Die folgenden beiden Zeilen drucken eine Datei, deren Name Leerzeichen enthält:

```
$ lpr 'Mein Tagebuch 1989.txt'
$ lpr Mein\ Tagebuch\ 1989.txt
```

Manchmal, besonders in der Shellprogrammierung, wünscht man sich, innerhalb eines Quotings Variablen verwenden zu können. Für solche Zwecke gibt es das unvollständige Quoting mit den doppelten Anführungszeichen:

```
" <Zeichenkette> "
```

Sinn machen diese Anführungszeichen beispielsweise, wenn man die Variable *$HOME* verwenden möchte:

```
$ echo "$HOME/tagebuch/Mein Tagebuch 1989.txt"
/home/quark/tagebuch/Mein Tagebuch 1989.txt
$ lpr "$HOME/tagebuch/Mein Tagebuch 1989.txt"
```

Möchte man Quoting-Zeichen innerhalb eines Quotings verwenden, muß man die Prioritäten der Shell kennen: Am stärksten binden die einfachen Anführungszeichen, dann der Rückstrich und schließlich die doppelten Anführungszeichen:

```
$ echo '\"Quoting\" ist ganz leicht'
\"Quoting\" ist ganz leicht
$ echo '"Quoting" ist ganz leicht'
"Quoting" ist ganz leicht
```

10.2.8 Verzeichnisstack

Die Bash enthält einen sogenannten Verzeichnisstack. Mit ihm ist es an der Eingabeaufforderung möglich, zu den Verzeichnissen zurückzuspringen, in denen man bereits war. Einzige Voraussetzung: Um das Verzeichnis zu wechseln, muß man den Shellbefehl *pushd* verwenden:

```
pushd [Verzeichnis]
```

pushd legt das aktuelle Verzeichnis auf den Verzeichnisstack und wechselt in das angegebene Verzeichnis. Einen Stack (oder zu deutsch: Keller) kann man sich als Bücherstapel vorstellen. Wir nehmen an, daß dieser Bücherstapel so wackelig aufgetürmt ist, daß es unmöglich ist, andere Bücher als das obere wegzunehmen, ohne den Turm zum Umkippen zu bringen. Legt man also ein Buch auf den Stapel (*pushd*), dann kann man nur genau *dieses* Buch vom Stapel wieder entfernen. Möchte man ein Buch, das weiter unten liegt (also älter ist) vom Stapel nehmen, muß man zuerst alle Bücher von oben abtragen, bis man das gewünschte Exemplar erreicht hat. Der Stack hat also die Eigenschaft, daß man immer nur das »neueste« Element entfernen kann.

Kommen wir wieder zum Verzeichnisstack zurück. Wir können also mit *pushd* ein Verzeichnis auf den Stack (den Bücherstapel) legen. Doch wie entfernt man ein Verzeichnis wieder vom Stack, um wieder in ein früheres Verzeichnis zu wechseln? Dies geschieht mit *popd*. *popd* nimmt das oberste Verzeichnis vom Stack und wechselt dort hinein. Den Inhalt des Verzeichnisstacks kann man sich mit dem Shellbefehl *dirs* ansehen:

```
$ pwd
/home/quark
$ pushd bin
~/bin ~
$ pushd /
/ ~/bin ~
$ dirs
/ ~/bin ~
$ dirs -l                                  # Tilde auflösen
/ /home/quark/bin /home/quark
$ popd
~/bin ~
$ pwd
/home/quark/bin
$ popd
~
$ pwd
/home/quark
```

Damit man zu beliebigen Verzeichnissen im Stack wechseln kann, gibt es bei *pushd* die Möglichkeit, den Stack zu rotieren. Möchte man in ein Verzeichnis, das sich bei der Ausgabe von *dirs* an beispielsweise dritter Position von links befindet, muß man *pushd* +2 eingeben (bei 0 zu zählen beginnen, deshalb +2 und nicht +3.):

```
$ dirs
/ ~/buch ~
$ pushd +2
~ / ~/buch
```

Linksherum läßt sich mit vorangestelltem Minuszeichen rotieren:

```
$ dirs
/ ~/buch ~
$ pushd -0
~/buch ~ /
```

pushd ohne Argumente wechselt zwischen den letzten beiden Verzeichnissen hin und her:

```
$ dirs
/ ~/buch ~
$ pushd
~/buch / ~
$ pushd
/ ~/buch ~
```

Mit den Optionen *+{n}* und *-{n}* von *popd* kann man Verzeichnisse vom Stack nehmen, ohne in diese zu wechseln. *popd +{n}* entfernt analog zu *pushd* das n-te Verzeichnis von links und *popd -{n}* das n-te von rechts:

```
$ pwd
/
$ dirs
/ ~/buch ~
$ popd +1
/ ~
$ popd -0
/
$ pwd
/
```

10.3 Dateinamenexpandierung

Pfade sind in Unix oft sehr lang. Um sich diese Tipparbeit zu ersparen, hat sich die Shell einiges einfallen lassen.

10.3.1 Jokerzeichen (Wildcards)

Ein Computer kann seine Stärke erst dann ausspielen, wenn er größere Aufgaben erledigen darf. Nur dann ist er in der Regel schneller als der Mensch. Besonders gut ist er bei stumpfsinnigen Betätigungen, die dem Menschen lästige Tipparbeit ersparen. Nehmen wir beispielsweise an, wir besitzen ein Verzeichnis mit 1024 Dateien und wollen alle in ein anderes Verzeichnis verschieben. Da ich annehme, daß es wenig Menschen auf dieser Welt gibt, die freiwillig 1024 Dateinamen eintippen, liegt es nahe, diese Aufgabe dem Rechenknecht zu überlassen. Dafür hat die Shell mit den Jokerzeichen (engl. Wildcards) vorgesorgt:

```
$ mv * /neues/verzeichnis
```

Jokerzeichen sind die Platzhalter in Suchmustern für Dateien. Der hier verwendete Stern »*« ist Platzhalter für beliebig viele Zeichen in einem Dateinamen. Besteht ein Suchmuster also nur aus »*«, entsprechen natürlich alle Dateien diesem Kriterium. Kleiner wird die Auswahl jedoch, wenn man folgenden Aufruf tätigt:

```
$ mv a* /neues/verzeichnis
```

Hier werden nur alle Dateien verschoben, deren Dateiname mit »a« beginnt. Das Suchmuster gibt also an, daß zu Beginn auf jeden Fall ein »a« verlangt wird, alle weiteren Zeichen können jedoch beliebig sein und beliebig oft vorkommen. Dementsprechend verschiebt folgendes Beispiel nur Dateien, deren Name mit »a« beginnt und mit »h« endet:

```
$ mv a*h /neues/verzeichnis
```

»*« allein findet alle Dateien bis auf die »versteckten« Dateien. Dies sind alle Dateien, deren Dateiname mit einem Punkt ».« beginnt. Um diese zu erreichen, muß der Punkt explizit angegeben werden. Folgender Aufruf löscht tatsächlich alle Dateien im aktuellen Verzeichnis:

```
$ rm .* *
```

Der Stern ist nicht das einzige gültige Jokerzeichen. Folgende Aufstellung bietet eine Übersicht über die Wildcards, die in jeder Shell verfügbar sind:

? genau ein Zeichen

Beispiele:

```
$ mkdir test
$ cd test
$ touch abba otto anna
$ echo a*a
abba anna
$ echo a?a

$ echo a??a
abba anna
```

[...] genau eines der innerhalb der eckigen Klammern stehenden Zeichen

Es können auch Zeichenbereiche angegeben werden, beispielsweise [a-z] für alle Zeichen von »a« bis »z« oder [0-9], für alle Zeichen von »0« bis »9«.

Beispiele:

```
$ mkdir test
$ cd test
$ touch ANNA anna
$ echo [a-z]nna
anna
$ echo [abc]nna
anna
$ echo [A-Z]*
ANNA
```

[!...] genau ein Zeichen, nur nicht die Zeichen innerhalb der Klammern (nicht in der C-Shell verfügbar)

Beispiele:

```
$ mkdir test
$ cd test
$ touch ANNA anna
$ echo [!a-z]*
ANNA
$ echo [!A]*
anna
```

Die Abarbeitung der Jokerzeichen nennt man *Dateinamenexpandierung*. Dieser treffende Begriff macht auf die Art und Weise, wie die Shell mit diesen Suchmustern umgeht, aufmerksam: Trifft die Shell auf ein Suchmuster, ersetzt sie es durch die Dateinamen, die auf das Suchmuster passen. All dies geschieht noch vor der Aufspaltung der Zeile in die einzelnen Wörter. Jeder zutreffende Dateiname wird also zu genau einem Wort. Diese Tatsache ist wichtig beim Umgang mit Dateinamen, die Leerzeichen enthalten.

10.3.2 Tilde-Expandierung

Einen wichtigen Beitrag zum Komfort, den die C-Shell einführte, leistet die Tilde-Expandierung: Mit ihr können wichtige Verzeichnisse mit sehr wenigen Tastendrücken erreicht werden.

Eine Tilde-Expandierung beginnt immer mit einem Tilde-Zeichen. Was die Shell anstelle des Tilde-Ausdrucks einsetzt, hängt nur davon ab, was nach der Tilde folgt. Folgende Ausdrücke sind in jeder Shell, die nach der C/Shell kam, gültig:

Ausdruck	wird ersetzt durch
~	das Homeverzeichnis ($HOME)
~+	das aktuelle Verzeichnis ($PWD)
~-	das zuletzt besuchte Verzeichnis ($OLDPWD)
~login	das Homeverzeichnis des angegebenen Benutzers

Kann ein Tilde-Ausdruck nicht aufgelöst werden, wird er als solcher belassen. Betrachten wir dazu ein paar Beispiele:

```
$ pwd
/usr/bin
$ cd ~/skripte
$ pwd
/home/quark/skripte
$ cd ~-/X11
$ pwd
/usr/bin/X11
$ cd ~odo
$ pwd
/home/odo
$ echo ~+
/home/odo
```

Ein Tilde-Ausdruck endet normalerweise beim nächsten Trennzeichen (Leerzeichen, Tabulator, neue Zeile). Damit aber auch bei Wertzuweisungen an Variablen wie *CDPATH* oder *PATH* die Tilde-Expandierung verwendet werden kann, wird ab der C-Shell auch der Doppelpunkt als Trennzeichen akzeptiert:

```
$ echo $PATH
/bin:/usr/bin:/home/quark/bin:.
$ PATH=/bin:/usr/bin:~/bin:.
$ echo $PATH
/bin:/usr/bin:/home/quark/bin:.
```

In der Bash gibt es noch weitere Tilde-Expandierungen, über die auf den Verzeichnisstack zugegriffen werden kann.

Ausdruck	wird ersetzt durch
~n	den Eintrag mit Position +n auf dem Verzeichnisstack
~+n	den Eintrag mit Position +n auf dem Verzeichnisstack
~-n	den Eintrag mit Position -n (rückwärts) auf dem Verzeichnisstack

10.3.3 Brace-Expandierung

Für diese Form der Expandierung, die in der C-Shell und der Bash möglich ist und *Brace Expansion* genannt wird, gibt es leider keinen passenden deutschen Ausdruck, deshalb hat sich auch in Deutschland der Begriff Brace-Expandierung durchgesetzt. Die Brace-Expandierung ist streng genommen kein Teil der Dateinamenexpandierung, da sie im Gegensatz zu den Jokerzeichen nicht dazu dient, vorhandene Dateien mit einem Suchmuster zu erreichen. Vielmehr *erzeugt* sie Dateinamen. Die allgemeine Syntax ist:

```
vorderteil{wert1,wert2,wert3,...}schlußteil
```

Man kann sich die Braces als Wiederholungsklammern vorstellen. Sie helfen dabei, unnötige Wiederholungen zu vermeiden und nur die Unterschiede zusammenzufassen. Nehmen wir an, wir möchten im Verzeichnis »~/testdateien« drei leere Dateien mit den Namen »a«, »b« und »c« erstellen. Der Befehl *touch* erstellt für uns Dateien mit 0 Byte. Normalerweise müßten wir also an der Eingabeaufforderung folgendes Kommando eingeben:

```
$ touch ~/testdateien/a ~/testdateien/b ~/testdateien/c
```

Diese Methode ist jedoch sehr ineffektiv, da wir dreimal dasselbe Verzeichnis »~/testdateien« eingeben müssen. Mit den Braces können wir die Pfade zusammenfassen:

```
$ touch ~/testdateien/{a,b,c}
```

Bei der Auflösung des Kommandos durch die Shell werden daraus wieder die drei Dateinamen:

```
$ touch ~/testdateien/a ~/testdateien/b ~/testdateien/c
```

Braces können beliebig geschachtelt werden. Möchten Sie zum Beispiel eine Datei in mehreren Dateien suchen, kann folgender geschachtelter Brace-Ausdruck helfen:

```
$ ls -CF /{bin,usr/{X11/bin,bin}}/y*
/bin/ypdomainname@     /usr/bin/yes*
/usr/bin/ypcat*        /usr/bin/ypmatch*
/usr/bin/yppasswd*     /usr/bin/ypwhich*
/usr/bin/ytalk*
```

Die Werte innerhalb der geschweiften Klammern werden erst ausgewertet, nachdem die Braces aufgelöst wurden. Damit ist es möglich, auch Suchmuster der Dateinamenexpandierung zu benutzen:

```
$ echo {~/texte/kapitel1,~/files/texte/*}
/home/quark/texte/kapitel1 /home/quark/files/texte/1
/home/quark/files/texte/2 /home/quark/files/texte/3
```

10.4 Ein- und Ausgabeverarbeitung

In diesem Kapitel erfahren Sie, wie man mit Eingabe- und Ausgabe von Programmen in NetBSD umgeht und welche Hilfsmittel die Shell dafür bietet. Wenn nicht anders angegeben, gelten die folgenden Ausführungen für alle besprochenen Shells.

10.4.1 Umlenkung

Die Ausgabe eines Programms rast so schnell über den Bildschirm, daß man die Zeichen nicht mehr lesen kann? Mit Hilfe der Umlenkung ist es möglich, den Text in einer Datei zu speichern, um sie dann weiterzuverarbeiten oder sie auszudrucken. Sie möchten eine Datei einem Programm zur Weiterverarbeitung übergeben? Auch hier hilft die Umlenkung.

Beginnen wir mit einem Beispiel: Wir möchten eine Liste mit allen via *pkg_add* installierten Programmen ausdrucken. Dazu speichern wir zuerst die Rückgabe von *ls* in einer Datei und drucken diese dann aus. Die dafür notwendige Umlenkung erreichen wir mit Hilfe von »>«:

```
$ cd /usr/pkg/bin; ls > $HOME/liste; lpr $HOME/liste
```

Das Größer-Zeichen sorgt also dafür, daß die Ausgabe des links stehenden Kommandos in die rechts stehende Datei geschrieben wird:

```
kommando > datei
```

Fehlermeldungen bleiben allerdings außen vor, sie werden weiterhin auf den Bildschirm geschrieben. Dies liegt an der Tatsache, daß es in Unix-Systemen wie NetBSD einen speziellen »Kanal« für solche wichtigen Meldungen gibt. Man kann sich diesen Mechanismus wie ein Röhrensystem vorstellen. NetBSD stellt der Shell drei Röhren zur Verfügung: Eine für die Standardeingabe (Tastatur), eine für die Standardausgabe (Bild-

schirm) und eine für die Fehlermeldungen (Bildschirm). Mit dem Größer-Zeichen lenken wir lediglich die Röhre für die Standardausgaben an ein anderes Ziel um, die Röhre für die Fehlermeldungen ist weiterhin mit dem Bildschirm verbunden. Natürlich bietet die Shell auch eine Möglichkeit, andere Röhren zu verlegen. Die Fehlermeldungen kann man durch Voranstellen einer »2« umlenken:

```
$ ls > liste 2> fehlermeldungen
```

Die »2« entspricht der sogenannten Kanalnummer der Standardfehlerausgabe, »1« steht für die Standardausgabe und »0« für die Standardeingabe.

In dem Beispiel landen die Dateinamen in der Datei »liste« und die Fehlermeldungen in der Datei »fehlermeldungen«. Sehr häufig lenkt man die Fehlermeldungen auch in den virtuellen Mülleimer »/dev/null«:

```
$ ls > liste 2> /dev/null
```

Um sowohl die Standardausgabe als auch die Standardfehlerausgabe umzulenken, kann man in allen Shells außer der Bourne-Shell die Umlenkung »>&« benutzen:

```
$ ls datei_die_nicht_existiert >& liste_und_fehler
$ cat liste_und_fehler
ls: datei_die_nicht_exisitiert: No such file or directory
```

Beachten Sie, daß die Shell bei der Umlenkung eventuell bereits bestehende Dateien ohne Vorwarnung überschreibt. Glücklicherweise gibt es jedoch in der C-Shell die Möglichkeit, dies durch Setzen der Variable *noclobber* abzuschalten:

```
$ set noclobber
```

In der Korn-Shell und der Bash muß man folgendes Kommando eingeben:

```
$ set -o noclobber
```

Versucht man nun, die Ausgabe in eine bestehende Datei zu schreiben, lehnt dies die Shell mit einer Fehlermeldung ab, beispielsweise meldet die Bash:

```
$ set -o noclobber
$ ls > liste
bash: liste: cannot overwrite existing file
```

Wünschen Sie trotz gesetzter Option *noclobber* die Datei zu ersetzen, gibt es mit »>|« in der Korn-Shell und der Bash und mit »>!« in der C-Shell Umlenkungspfeile, die dies umgehen:

```
$ ls > liste
bash: liste: cannot overwrite existing file
$ ls >| liste
$ ...
```

Oft möchte man eine Datei nicht überschreiben, sondern um einige Zeilen erweitern. Dies ist mit der Umlenkung »>>« möglich. Sie hängt die Ausgabe eines Kommandos an die bestehende Datei an:

```
$ echo Zeile 1 > test
$ echo Zeile 2 >> test
$ cat test
Zeile 1
Zeile 2
```

Neben Standardausgabe und Standardfehlerausgabe kann man natürlich auch die Standardeingabe umlenken. Dies geschieht mit »<«. Viele Programme in Unix arbeiten wie ein Filter. Ruft man sie ohne einen Parameter auf, lesen sie ihre Daten von der Standardeingabe ein. Beispielsweise übergibt folgender Befehl die Datei »/etc/motd« dem Unixtool *wc*, das Zeichen, Wörter und Zeilen zählt:

```
$ wc < /etc/motd
       4        13        78
```

10.4.2 Pipelines

Mit der Umlenkung ist es möglich, die bereits besprochenen Kanäle in Dateien oder auch Geräte wie »/dev/null« umzulenken. Die Verbindung zwischen zwei Programmen können sie jedoch nicht herstellen. Dafür gibt es die Pipeline:

```
<Kommando1> | <Kommando2> | [...]
```

Eine Pipe verbindet die Standardausgabe des ersten Kommandos mit der Standardeingabe des zweiten Kommandos und die Standardausgabe des zweiten Kommandos mit der Standardeingabe des dritten und so weiter.

Sehr häufig wird die Pipe in Verbindung mit dem Seitenbetrachter *more* verwendet:

```
$ ls | more
ls: Permission denied
```

Wie man an diesem Beispiel sieht, wird selbstverständlich die Standardfehlerausgabe nicht umgelenkt, Fehlermeldungen landen also weiterhin auf dem Bildschirm.

Gäbe es die Pipe nicht, müßten wir das vorherige Beispiel folgendermaßen angehen:

```
$ ls > tmp
$ more < tmp
$ rm tmp
```

10.4.3 Kommandosubstitution

Die Kommandosubstitution ist ein mächtiges Instrument der Shell. Mit ihr ist es möglich, die Standardausgabe eines Kommandos beispielsweise als Parameter für ein anderes Kommando zu benutzen. Hierfür hat die Shell die Rückstriche (engl. Backticks) vorgesehen:

```
` <Kommando> `
```

In allen Shells außer der Bourne-Shell ist auch noch die wesentlich einfachere und zu favorisierende »neue Syntax« möglich:

```
$( <Kommando> )
```

Nehmen wir beispielsweise an, wir möchten alle Dateien löschen, die auf ».o« enden – im aktuellen Verzeichnis und dessen Unterverzeichnissen. Für die Suche nach Dateien in Verzeichnisbäumen ist das Unixtool *find* zuständig:

```
$ find . -name \*.o
./main.o
./tools/viewer.o
```

Diese Liste übergeben wir nun *rm* mit Hilfe einer Kommandosubstitution:

```
$ rm `find . -name \*`
```

oder

```
$ rm $(find . -name \*)
```

Vielleicht fragen Sie sich, warum die Kommandosubstitution aus den zwei Zeilen zwei Parameter macht. Dies hängt mit der bereits besprochenen Variable *IFS* zusammen. Sie enthält alle Zeichen, die die Shell als Trennzeichen erkennen soll. Standardmäßig sind dies das Leerzeichen, der Tabulator und das Neue-Zeile-Zeichen (engl. newline). Da find nach jedem Dateinamen ein Neue-Zeile-Zeichen ausgibt, kann die Shell dies also als ein Trennzeichen verstehen.

Kommandosubstitutionen können beliebig geschachtelt werden. Allerdings wird die Syntax – zumindest bei Verwendung von Rückstrichen – bei zunehmender Verschachtelungstiefe immer schwieriger, da die inneren Backticks gequotet werden müssen. Wesentlich einfacher ist in solchen Fällen die »neue Syntax«, bei der kein Quoting notwendig ist:

```
$ cat a
Dringlichkeit besteht immer
$ cat b
a
$ echo $(cat $(cat b))
Dringlichkeit besteht immer
```

10.4.4 Überlange Kommandos (xargs)

Suchten Sie schon einmal verzweifelt nach einer Datei auf Ihrem System? Und gingen Sie auch schon mal so weit, jede Datei auf dem System mit

```
$ grep wasichsuche `find -name "*" -print`
```

nach dem gewünschten Suchwort zu durchsuchen? Wenn ja, dann ist die Wahrscheinlichkeit sehr hoch, daß auch Sie irgendwann eine Fehlermeldung wie

```
arg list too long
```

erhielten. Das Problem liegt darin, daß NetBSD nur einen festen Speicherbereich für die Argumente eines Programms zur Verfügung stellt. Überschreitet die Argumentenliste diese Größe, erhält man eine Fehlermeldung. Zur Lösung dieses Problems gibt es das Unixtool *xargs*:

```
xargs [-...] [Kommando]
```

Die Argumente für das angegebene Kommando erwartet *xargs* an der Standardeingabe. Als Trennzeichen zwischen den Argumenten gelten das Leerzeichen, der Tabulator und das Neue-Zeile-Zeichen, dabei sind Quotings zur Zusammenfassung von Dateinamen mit Leerzeichen wie in der Shell erlaubt (»""«, »'« und Backslash). Damit klar wird, wie *xargs* zu verwenden ist, betrachten wir am besten die für das Beispiel vom Anfang des Kapitels geeignete Lösung:

```
$ find -name "*" -print | xargs grep wasichsuche
```

Als erstes fällt auf, daß aus der Kommandosubstitution eine Pipe wurde. Die Dateinamen, die *find* ausgibt, landen an der Standardeingabe von *xargs* – das Limit der Kommandozeile spielt somit keine Rolle mehr. *xargs* liest nun soviele Argumente ein, wie auf

dem aktuellen System in eine Argumentenliste passen, und ruft das angegebene Kommando mit den gesammelten Parametern auf. Dieser Zyklus von »Argumente einlesen« und »Kommando aufrufen« wird solange wiederholt, bis entweder das EOF-Zeichen (End of File) erreicht ist oder eine Zeile gefunden wird, die nur aus einem einzigen Unterstrich (»_«) besteht.

Nun kann man sich vorstellen, wo der große Vorteil von *xargs* gegenüber anderen manuellen Methoden wie

```
$ find -name "*" -print |
> while read f; do grep wasichsuche "$f"; done
```

liegt. *xargs* minimiert die Anzahl der Aufrufe eines externen Programms auf das absolut notwendige. Ein einfacher Vergleich belegt diese Behauptung sehr deutlich:

```
$ find -name "*" -print |
> (while read f; do grep wasichsuche "$f"; done; times)
...
0m1.76s 0m4.66s
0m19.82s 0m41.82s

$ (find -name "*" -print | xargs grep wasichsuche; times)
...
0m0.01s 0m0.01s
0m3.13s 0m31.46ms
```

Auf die Bedeutung der ausgegebenen Zeitwerte möchte ich nicht genauer eingehen. Auffallend ist jedoch auch ohne Vorkenntnisse, daß offensichtlich die Zeile mit *xargs* wesentlich schneller abgearbeitet wird.

10.5 Aliase

Unser grundsätzliches Ziel sollte sein, Wege und Möglichkeiten zu finden, die Shellumgebung so angenehm wie möglich zu gestalten, so daß wir uns wie zu Hause fühlen. Einen ersten Schritt in die richtige Richtung machen wir mit den Aliasen, die in jeder Shell außer der Bourne-Shell möglich sind.

10.5.1 Aliase definieren

Die meiste Zeit verbringt ein Shell-Benutzer mit Wiederholungen. Wie oft haben Sie bereits *ls -l* eingegeben? Wie oft *tar zxvf*? Um die Eingabe solcher immer wiederkehrender, gleichlautender Kommandos zu vereinfachen, gibt es Aliase, die vergleichbar mit Abkürzungen sind. In der C-Shell schreibt man:

```
alias <Alias> <Kommando>
```

In der Korn-Shell und der Bash gilt hingegen die folgende Syntax:

```
alias <Alias>=<Kommando>
```

Während ein Alias nur ein Wort sein darf, kann das Kommando, für das es steht, aus beliebig vielen Wörtern bestehen – in der Praxis ist dies meistens der Fall. Einzige Bedingung ist jedoch, daß das Kommando gequotet wird:

```
$ alias m=more
$ alias l='ls -l'
$ alias t='tar zxvf'
$ alias xxx='echo Alias xxx wurde benutzt:'
$ xxx Es fährt kein Zug nach Eierbach
Alias xxx wurde benutzt: Es fährt kein Zug nach Eierbach
```

Da Aliase nur den Anfang eines Kommanods abkürzen können, sind sie freilich kein Allheilmittel. Sobald variable Parameter innerhalb des Kommandos, das durch ein Alias abgekürzt werden soll, benötigt werden würden, muß das Alias passen. Beispielsweise können Sie folgende Zeile nicht durch ein Alias abkürzen:

```
dd if=<Gerät> of=<Datei>
```

dd ist ein Unixtool, das Daten von beliebigen Quellen, also auch Geräten, in ein beliebiges Ziel Byte für Byte kopiert und dabei auf Wunsch einige Konvertierungen vornimmt. Sehr häufig dient es der Erstellung eines CD-Images:

```
$ dd if=/dev/acd0 of=image.iso
```

Dieses Beispiel stellt uns sogar vor zwei Probleme: Zum einen können wir den Parameter *<Gerät>* mit einem Alias nicht umsetzen und zum anderen darf sich vor und nach den Gleichheitszeichen kein Leerzeichen befinden. Deshalb können wir nicht einmal folgendes Alias definieren:

```
$ alias d='dd if=/dev/acd0 of='
$ d image.iso
dd: no value specified for of
```

Beachten Sie bitte: Da Aliase nur in der jeweils aktuellen Sitzung (Shellinstanz) gelten, müssen sie zur dauerhaften Benutzung in einem Startskript definiert werden. Ein Alias ist also tatsächlich nur ein sehr einfaches Werkzeug, das allerdings blitzschnell zur Verfügung steht. Wesentlich komplexere Aufgaben können mit sogenannten Shellskripten automatisiert werden.

10.5.2 Aliase anzeigen

Eine Liste aller definierten Aliase liefert der Aufruf von *alias* ohne Parameter:

```
$ alias
m=more
l='ls -l'
t='tar zxvf'
```

Übergibt man den Namen eines Aliases, wird nur dessen Definition angezeigt:

```
$ alias t
t='tar zxvf'
```

Sehr häufig definiert man »auf die Schnelle« ein Alias und bemerkt erst später, daß man es noch häufiger benötigen könnte. Manuell müßte man nun in das Startskript gehen und die entsprechende Zeile mit der Aliasdefinition nochmals eingeben. Die Bash erspart uns diese Tipparbeit, wenn wir *alias* folgendermaßen aufrufen:

```
$ alias -p m
alias m='more'
```

Diese Ausgabe kann ohne Veränderung in einem Startskript eingefügt werden:

```
$ alias -p m >> .bashrc
```

10.5.3 Aliase umgehen und löschen

Aliase sind sehr unauffällige Helfer, die auch dann einsetzen, wenn man sie nicht verwenden will. Oft vergißt man sie sogar, so daß sie schon für die eine oder andere Überraschung sorgen. Deshalb muß es einen Weg geben, Aliase zumindest temporär zu umgehen. Hierfür bietet sich das Quoting an:

```
$ alias echo='echo Alias wurde benutzt:'
$ echo Hallo Welt
Alias wurde benutzt: Hallo Welt
$ \echo Hallo Welt
Hallo Welt
```

Um die Funktionsweise dieses Tricks zu verstehen, muß man wissen, daß die Shell Aliase noch vor dem Quoting abarbeitet. Das Quoting ist also nicht nicht aufgelöst, wenn die Shell nach Aliasen sucht. Benötigt man ein Alias in dieser Sitzung überhaupt nicht mehr, kann man es selbstverständlich auch löschen. Dazu dient *unalias <Aliase>:*

```
$ alias echo='echo Alias wurde benutzt:'
$ echo Hallo Welt
Alias wurde benutzt: Hallo Welt
$ unalias echo
$ echo Hallo Welt
Hallo Welt
```

10.5.4 Alternativen

Da die Bourne-Shell keine Aliase kennt, müssen wir dort eine etwas umständlichere Alternative einsetzen: Shellskripte. Ein Shellskript ist eine Textdatei, die Befehle enthält, die die Shell so ausführt, als gäbe man sie nacheinander an der Kommandozeile ein.

Betrachten wir folgendes Alias:

```
$ alias t='tar zxvf'
```

Das funktional gleichwertige Shellskript sieht folgendermaßen aus:

```
$ cat t
#!/bin/sh
tar zxvf "$@"
```

Die erste Zeile teilt dem Betriebssystem mit, daß wir dieses Skript mit der Bourne-Shell ausführen möchten. Die zweite Zeile enthält das Kommando und einen Platzhalter, der alle an der Kommandozeile übergebenen Parameter an *tar* weitergibt. Den ersten Parameter allein erreicht man übrigens mit *$1*, den zweiten mit *$2* und so weiter. Um dieses Skript lauffähig zu machen, müssen wir zwei Schritte ausführen:

1. Wir erstellen ein Verzeichnis »$HOME/bin« und fügen es in den *PATH* ein:

   ```
   $ export PATH=$HOME/bin:$PATH
   ```

2. Wir müssen das Recht besitzen, das Skript ausführen zu können, also rufen wir *chmod* auf:

   ```
   $ chmod u+x t
   ```

Eine ganz andere Möglichkeit besteht, wenn man Variablen als Aliase benutzt. Beispielsweise könnte unser Beispiel folgendermaßen realisiert werden:

```
$ t='tar zxvf'
$ $t backup.tgz
```

Sie sehen schon, daß Aliase weitaus einfacher zu handhaben sind. Ein weiterer Grund, die Bourne-Shell nicht als Login-Shell einzusetzen...

10.6 Die History

Aliase sind sehr praktisch, müssen jedoch erst definiert werden. Falls sich diese Zusatzarbeit nicht lohnt, gibt es in allen Shells außer der Bourne-Shell eine History-Funktion, die dem Anwender ermöglicht, zuvor eingegebene Kommandos mit wenigen Tastendrücken zu wiederholen.

10.6.1 Die History in der C-Shell

Die History-Funktion der C-Shell war die erste ihrer Art und ist deshalb für heutige Verhältnisse nicht ausreichend, für die damalige Zeit jedoch sensationell. Die wesentlichen Mechanismen beschränken sich auf die traditionellen Methoden der Shell: Substitution. Die Shell verbringt viel Zeit mit der Ersetzung von Ausdrücken, beispielsweise Variablen- oder Kommandosubstitutionen. Hinzu kommt in diesem Kapitel die History-Substitution. Beginnen wir jedoch mit der einfachsten Möglichkeit, dem Anzeigen aller bereits eingegebenen Kommandos:

```
$ history
     1  echo Hallo History
     2  cd
     3  pwd
     4  tar czvf /backup/backup.tar.gz
     5  history
```

Einfache History-Funktionen

Um ein Kommando zu wiederholen, können wir uns der History-Substitution bedienen. Jeder Ausdruck dieser Art beginnt mit einem Ausrufezeichen und ist gefolgt durch ein bestimmendes Suchmuster. Das einfachste ist die History-Nummer:

```
$ !3
pwd
/home/quark
```

Damit wird der dritte Eintrag in der Historyliste wiederholt. Bevor die Shell das Kommando ausführt, schreibt es die zuvor eingegebene Zeile nochmals auf die Standardausgabe. Wie viele Einträge in der History gespeichert werden, legt man mit der Variable *history* fest. Erreicht die History diese Größe, wird das älteste Element gelöscht. Standardmäßig ist die Variable *history* nicht gesetzt, was bedeutet, daß die Größe theoretisch unbeschränkt ist:

```
$ set history=100
```

Es kann auf History-Einträge auch relativ zur aktuellen Position zugegriffen werden. Beispielsweise wiederholt folgendes Beispiel die Zeile, die dem Ausdruck »!3« entspricht:

```
$ !-4
pwd
/home/quark
```

Beachten Sie, daß auch Rückgriffe auf die History wiederum in der History landen, wie an diesem Beispiel ersichtlich war!

Für den häufig vorkommenden Fall, daß man die zuletzt eingegebene Zeile wiederholen möchte, gibt es die Abkürzung »!!«:

```
$ echo Paper Tiger
Paper Tiger
$ !!
echo Paper Tiger
Paper Tiger
```

Die History-Substitution gilt nicht nur zu Beginn einer Zeile, sie kann an beliebiger Stelle verwendet werden. Beispielsweise versucht folgendes Beispiel ein Skript auszuführen, das offensichtlich nicht vorhanden ist und erstellt dann diese Datei mit einem Editor:

```
$ cd $HOME/bin
$ ./start_x
csh: ./start_x: No such file or directory
$ vi !!
vi ./start_x
```

Anstatt der unhandlichen History-Nummer kann man auch den Beginn eines History-Eintrags angeben:

```
$ !p
pwd
/home/quark/bin
```

Falls die Suche nicht eindeutig ist, wird der neueste passende History-Eintrag gewählt:

```
$ vi start_x
...
$ vi backup
...
$ !vi
vi backup
```

Wir können auch in der Mitte der History-Einträge suchen:

```
$ !?w?
pwd
/home/quark/bin
```

Teile aus History-Einträgen ausschneiden

Für den Power-User gibt es noch einige sehr raffinierte Substitutionen, die ihm erlauben, nur einzelne Wörter beziehungsweise Parameter von einem History-Eintrag zu übernehmen. Diese Anweisungen werden alle mit einem Doppelpunkt an eine History-Substitution angehängt. Beispielsweise verwendet folgendes Beispiel den ersten Parameter des zuletzt eingegebenen Kommandos:

```
$ cc main.c -o super_c_programm
main.c: In function `main':
main.c:3: `i' undeclared
$ vi !!:1
vi main.c
```

Die folgende Tabelle liefert eine Übersicht:

Ausdruck	entspricht
0	Kommando
n	n-ter Parameter
^	erster Parameter
$	letzter Parameter
%	Wort, das dem vorher angegebenen Suchmuster der Form »?s?« entspricht
x-y	Parameter von x bis y
-y	das gleiche wie 0-y
*	alle Wörter (dasselbe wie »^-$)«
x*	alle Wörter von x bis zum letzten (dasselbe wie »x-$)«
x-	alle Wörter von x bis ausschließlich dem letzten

Betrachten wir dazu ein paar Beispiele:

```
$ echo Wort1 Wort2 Wort3 "Wort 4"
$ history | tail -1
     9  echo Wort1 Wort2 Wort3 "Wort 4"
```

```
$ echo !ech:%
echo echo
echo
$ echo !9:0
echo echo
echo
$ echo !9:4
echo "Wort 4"
Wort 4
$ echo !9:2-4
echo Wort2 Wort3 "Wort 4"
Wort2 Wort3 Wort 4
```

Modifizierer

Modifizierer gehen noch einen Schritt weiter, sie können jedes Resultat einer History-Substitution verändern. Einen Überblick über die wichtigsten Modifizierer liefert folgende Tabelle:

Modifizierer	Beschreibung	Beispiel
h	liefert nur den Pfad	*/usr/bin/vi => /usr/bin*
r	entfernt Dateinamenendung	*buch.txt => buch*
e	liefert nur die Dateinamenendung	*buch.txt => txt*
t	liefert den Dateinamen ohne Pfad	*/usr/bin/vi => vi*
s/l/r/	ersetzt l durch r	
&	wiederholt eine vorherige Substitution mit »s/l/r/«	

Modifizierer werden an einen History-Ausdruck mit einem Doppelpunkt angehängt:

```
$ vi buch.txt
...
$ !!:s/vi/more/
more buch.txt
...
$ echo !-2:1:e
echo txt
txt
```

Falls Sie vorsichtshalber die Wirkung einer History-Substitution testen möchten, können Sie den Modifizierer *p* benutzen, der dafür sorgt, daß die C-Shell das Kommando nicht ausführt, nur ausgibt:

```
$ vi buch.txt
...
$ !!:s/vi/more/:p
more buch.txt
$ ...
```

Obwohl die Modifizierer sehr mächtig sind, kann es vorkommen, daß ihr Gebrauch im Vergleich mit der manuellen Methode keinen Zeitvorteil bringt.

10.6.2 Die History in der Korn-Shell und der Bash

Während die C-Shell lediglich mit Hilfe von speziellen Shell-Ausdrücken bereits eingegebene Kommandos wiederholen kann, haben die Korn-Shell und die Bash eine weitaus bequemere Methode gefunden: Sie belegen Tastenkombinationen mit verschiedenen Funktionen, allen voran die berühmten Cursortasten, mit denen man in der History blättern kann. Die Bash ist auf diesem Gebiet abwärtskompatibel zur Korn-Shell *und* zur C-Shell, unterstützt also neben der History-Substitution auch einen Zeileneditor.

Zeileneditor

Um den Zeileneditor einzuschalten und damit auch die History-Funktionen auf die üblichen Tasten zu legen, muß das Kommando *set -o emacs* eingegeben werden.

Damit wird der sogenannte Emacs-Modus aktiviert, der die von dem gleichnamigen Editor bekannten Tastenkombinationen setzt. Auf die Alternative, den vi-Modus, soll an dieser Stelle nicht näher eingegangen werden, da kundige Anwender dessen Befehle sowieso kennen. Nun können wir beispielsweise mit [Cursor-Auf] zurück- und mit [Cursor-Ab] in der History vorblättern. Weitere wichtige Tastenkombinationen zeigt die folgenden Tabelle:

Tastenkombination	Beschreibung
[Strg]+[P]	springt zum vorherigen Eintrag in der History
[Strg]+[N]	springt zum nächsten Eintrag in der History
[Esc]+[<]	springt zum ersten Eintrag der History
[Esc]+[>]	springt zum letzten Eintrag der History
[Esc] *n* [Esc]+[g]	springt zum Eintrag *n* in der History
[Esc] *n* [Esc]+[.]	setzt an der aktuellen Cursorposition den *n*-ten Parameter des vorherigen Kommandos ein
[Esc]*n* [Esc]+[_]	setzt an der aktuellen Cursorposition den letzten Parameter des vorherigen Kommandos ein

Beachten Sie, daß [Esc] und die nachfolgende Taste jeweils hintereinander gedrückt werden müssen.

Eine weitere sehr hilfreiche Funktion ist die inkrementelle Suche. Nach Eingabe von [Strg]+[R] zeigt die Korn-Shell den zu den bereits eingegebenen Buchstaben passendsten Histroy-Eintrag an und sucht dabei auch innerhalb der Zeilen. Bei mehreren zutreffenden Zeilen wählt die Shell die neueste aus.

Wie Sie die Tastenbelegung des Zeileneditors verändern können, zeigt Kapitel 10.7.5.

Der Shellbefehl fc

Den Inhalt der History kann man in der Korn-Shell und der Bash wie in der C-Shell mit dem Befehl *history* betrachten. Ebenso wie die C-Shell kennen auch die Korn-Shell und die Bash eine Möglichkeit, mit der Eingabe eines Kommandos bereits eingegebene Zeilen wiederherzustellen. Anstatt der History-Substitution steht in diesen Shells der Shellbefehl *fc* zur Verfügung, der allerdings ähnliche Funktionen bietet:

```
fc [erster Eintrag] [letzter Eintrag]
```

fc können als Argumente History-Nummern oder auch Präfixe von History-Einträgen übergeben werden. Bevor die angegebenen Kommandos ausgeführt werden, ruft die Shell den Editor auf, der in der Variable *FCEDIT* angegeben ist, und bietet Ihnen die Möglichkeit, die Zeilen zu verändern. Da standardmäßig der spartanische »/bin/ed« verwendet wird, sollten Sie diese Variable sofort Ihren Wünschen anpassen, beispielsweise:

```
$ FCEDIT=/usr/bin/vi
$ history
1       ls
2       echo In My Place
3       ls -l
$ fc 2
...
echo In My Place
In My Place
$ fc e l
echo In My Place
ls -l
In My Place
...
```

Mit Hilfe des vordefinierten Aliases *r* kann das zuletzt eingegebene Kommando direkt, also ohne Editor, wiederholt werden:

```
$ echo Wiederholungen sind öde
Wiederholungen sind öde
$ r
echo Wiederholungen sind öde
Wiederholungen sind öde
```

Allgemein ist es natürlich mit *r* auch möglich, jedes andere Kommando direkt zu wiederholen:

```
$ r echo
echo Wiederholungen sind öde
Wiederholungen sind öde
```

Darüber hinaus besteht die Möglichkeit, Teile des wiederhergestellten Kommandos durch eine beliebige Zeichenkette zu ersetzen:

```
$ cc main.c -o super_c_programm
main.c: In function `main':
main.c:3: `i' undeclared
$ r "cc=cc -Wall"
cc -Wall main.c -o super_c_programm
```

10.7 Eingabe editieren

Seit der C-Shell besitzt jede Shell einen sogenannten Zeileneditor. Dies ist im Wesentlichen eine stark erweiterte und konfigurierbare Eingaberoutine, die den Komfort eines handelsüblichen Texteditors bietet. Beispielsweise ist es dort möglich, den Cursor an beliebiger Stelle in der Zeile zu positionieren. Die Bourne-Shell besitzt keine eigene Eingaberoutine und benutzt deshalb die des Betriebssystems, die außer [Backspace], [Strg]+[U] und [Enter] keine weiteren Tastenkombinationen kennt.

10.7.1 Deutsche Tastaturbelegung in der Bash

Die Bash ist in NetBSD überhaupt nicht konfiguriert und hat deshalb massive Probleme mit Umlauten. Abhilfe schafft eine Datei »/etc/inputrc« mit folgendem Inhalt:

```
set convert-meta off
set input-meta on
set output-meta on
```

Die Datei »/etc/inputrc« enthält alle Einstellungen des Zeileneditors der Bash. Mit dieser Konfiguration kann auch die Dateinamenexpandierung mit Umlauten umgehen. Die Schwächen des Unixtools */bin/ls* von NetBSD können damit jedoch natürlich nicht umschifft werden:

```
$ echo *
übel
$ ls ?bel
?bel
$ ls übel
übel
```

10.7.2 Die wichtigsten Tastenkombinationen

Für die alltägliche Arbeit mit der Shell muß man die wichtigsten Tastenkombinationen kennen, um effektiv vorgehen zu können. Dieses Kapitel liefert eine kleine Auswahl der in der C-Shell, der Korn-Shell und der Bash vorhandenen Tastenkombinationen. In der Korn-Shell muß der Zeileneditor mit

```
$ set -o emacs
```

eingeschaltet werden.

Cursor bewegen

Tastenkombination	Funktion	Beschreibung
[Strg]+[A] oder [Pos1]	beginning-of-line	springt an den Zeilenanfang
[Strg]+[E] oder [Ende]	end-of-line	springt an das Zeilenende.
[Strg]+[F] oder [Cursor-Rechts]	forward-char	ein Zeichen vorwärts
[Strg]+[B] oder [Cursor-Links]	backward-char	ein Zeichen rückwärts
[Meta]+[f]	forward-word	springt ein Wort vorwärts
[Meta]+[b]	backward-word	springt ein Wort zurück

Editieren

Tastenkombination	Funktion	Beschreibung
[Strg]+[D] oder [Entf]	delete-char	löscht das aktuelle Zeichen
[Strg]+[K]	kill-line	löscht den Rest der Zeile
[Meta]+[d]	kill-word	löscht den Rest des aktuellen Worts beziehungsweise löscht das nächste Wort, falls sich der Cursor zwischen zwei Wörtern befindet
[Meta]+[Entf]	backward-kill-word	löscht alle Zeichen bis zum Beginn des aktuellen Worts beziehungsweise löscht das vorherige Wort, falls sich der Cursor zwischen zwei Wörtern befindet
[Strg]+[y]	yank	fügt die zuletzt gelöschten Zeichen an der aktuellen Position ein

Bash-spezifische Tasten

In der Bash finden sich noch einige sehr hilfreiche Tastenkombinationen:

Tastenkombination	Funktion	Beschreibung
[Strg]+[_] oder [Strg-X][Strg-U]	undo	macht vorherige Änderung rückgängig (Undo)
[Meta]+[r]	revert-line	macht alle Änderungen in dieser Zeile rückgängig
[Strg]+[X] [*]	glob-expand-word	ersetzt ein Suchmuster der Dateinamenexpandierung durch die dazu passenden Dateinamen
[Strg]+[L]	clear-screen	löscht den Bildschirm und stellt die aktuelle Zeile wieder dar

Auch hier entspricht [Meta] der Taste [Esc] oder der Taste [Alt], falls Sie die amerikanische Tastenbelegung verwenden.

10.7.3 Vervollständigung

Der geübte Unix-Benutzer gibt bei der Arbeit an der Kommandozeile nur wenig vollständig ein, da er intensiv von dem sogenannten Completion-System Gebrauch macht und deshalb seiner Shell die ganze Arbeit überläßt. Was hat es mit dieser magischen Funktion auf sich? Am einfachsten und bekanntesten läßt dieser Mechanismus auf die

Tabulator-Taste der Bash reduzieren. Nehmen wir an, wir möchten die Datei »/etc/rc.local« mit dem vi editieren. Dazu geben wir in der Bash folgende Zeichen ohne nachfolgendes [Enter] ein:

```
$ vi /etc/rc.lo
```

Da die Bash für uns die restliche Tipparbeit übernehmen soll, drücken wir nun die Tabulator-Taste. Tatsächlich fügt die Shell die restlichen Zeichen »cal« an das Wort an, problematisch wird die Vervollständigung erst, falls unsere Eingabe nicht eindeutig ist, beispielsweise wenn wir folgendes eingeben:

```
$ vi /etc/rc[Tab]
```

Hier gibt die Bash einen Piepton aus, bei einem nochmaligen Drücken der Tabulatortaste zeigt sie uns dann die bisher passenden Dateinamen:

```
$ vi /etc/rc[Tab][Tab]
rc          rc.conf.d   rc.lkm      rc.shutdown
rc.conf     rc.d        rc.local    rc.subr
```

Diese Ausgabe zeigt eindeutig, daß es mehrere Dateinamen im Verzeichnis »/etc« gibt, die mit »rc« beginnen. Der Fehlerton sagt uns also, daß wir unsere Eingabe präzisieren müssen. Geben wir nun ».l« ein, reicht dies immer noch nicht, allerdings sind diesmal nicht mehr so viele Möglichkeiten übrig geblieben:

```
$ vi /etc/rc.l[Tab]
/etc/rc.lkm     /etc/rc.local
```

Die Zeilenvervollständigung ist in den Shells sehr unterschiedlich fortgeschritten: Die C-Shell kann lediglich Dateinamen vervollständigen, ebenso die Korn-Shell, die allerdings bereits eine konfigurierbare Tastenbelegung bietet. Die Bash kann alle diese Funktionen aufbieten und darüber hinaus auch noch Benutzernamen, Rechnernamen und Variablen vervollständigen. Da die Tastenkombinationen für die Zeilenvervollständigung leider nicht untereinander kompatibel sind, bietet folgende unvollständige Auflistung eine Übersicht:

C-Shell

Tastenkombination	Beschreibung
[Strg]+[D]	zeigt alle momentan passenden Dateinamen an
[Esc]	Vervollständigung von Dateinamen

Korn-Shell

In der Korn-Shell ist die Vervollständigung nur dann verfügbar, wenn der Zeileneditor mit

```
$ set -o emacs
```

eingeschaltet ist.

Tastenkombination	Funktion	Beschreibung
[Esc] [Esc]	complete	Vervollständigung von Dateinamen und Befehlen
[Strg]+[X] [Esc]	complete-commando	Vervollständigung von Befehlen
[Esc] [Strg]+[X]	complete-file	Vervollständigung von Dateinamen
[Esc] [=]	complete-list	zeigt alle momentan passenden Dateinamen und Kommandos an

Falls Sie die Standardeinstellung der Korn-Shell wie ich etwas unhandlich empfinden, sollten Sie Kapitel 10.11.4 lesen, das den Tabulator für die Vervollständigung konfiguriert.

Bash

Tastenkombination	Funktion	Beschreibung
[Esc] [Esc]	complete	Vervollständigung von Dateinamen und Kommandos
[Meta]+[!]	complete-commando	Vervollständigung von Befehlen
[Strg]+[X] [!]	possible-commando-completions	zeigt alle momentan passenden Befehle
[Meta]+[/]	complete-filename	Vervollständigung von Dateinamen
[Strg]+[X] [/]	possible-filename-completions	zeigt alle momentan passenden Dateinamen
[Meta]+[~]	complete-username	Vervollständigung von Benutzernamen

Tastenkombination	Funktion	Beschreibung
[Strg]+[X] [~]	possible-username-completions	zeigt alle momentan passenden Benutzernamen
[Meta]+[$]	complete-variable	Vervollständigung von Variablennamen
[Strg]+[X] [$]	possible-variable-completions	zeigt alle momentan passenden Variablennamen
[Meta]+[@]	complete-hostname	Vervollständigung von Rechnernamen
[Strg]+[X] [@]	possible-hostname-completions	zeigt alle momentan passenden Rechnernamen
[Meta]+[?]	possible-completions	zeigt, wie die Completion das aktuelle Wort vervollständigen könnte
[Meta]+[*]	insert-completions	fügt alle passenden Elemente der Completion ein
ohne Belegung	menu-complete	Vervollständigung wie bei complete, jedoch bei mehreren Möglichkeiten die erste einsetzen; beim nächsten Tastendruck das nächste und so weiter
[Meta]+[{]	complete-into-braces	fügt alle passenden Elemente der Completion ein und setzt sie in die geschweifte Klammern der Brace-Expandierung ein
[Meta]+[Tab]	dynamic-complete-history	Vervollständigung mit Elementen der History

10.7.4 Vervollständigung konfigurieren

Bei der Vervollständigung wünscht man sich oft, daß gewisse Dateien nicht berücksichtigt werden, da man sie meistens nicht benötigt. Ein klassisches Beispiel hierfür sind die Object-Dateien des C-Compilers. Wendet man beispielsweise auf folgende Zeile die Completion an, stellt die Shell mindestens zwei Dateien zur Wahl:

```
$ vi main[Tab]
main.c    main.o
```

Um Dateien mit solchen unerwünschten Endungen von der Completion auszuschließen, gibt es die Variable *fignore* in der C-Shell beziehungsweise *FIGNORE* in der Bash. In der C-Shell sieht ein beispielhafter Aufruf folgendermaßen aus:

```
$ set fignore=(.o ~ .bak)
```

In der Bash müssen die Endungen mit Doppelpunkten getrennt werden:

```
$ FIGNORE='.o:~:.bak'
```

Completion-System der Bash konfigurieren

Ein grundsätzliches Problem der Vervollständigung ist die Fixierung auf Dateien. Auch wenn die Bash das Tätigkeitsfeld etwas erweitert hat, bleibt die Tatsache, daß die Vervollständigung »dumm« ist. Nehmen wir als Beispiel an, wir möchten eine Postscript-Datei mit Ghostview betrachten:

```
$ gv [Tab][Tab]
main.c   doku.ps
```

Wie gut wäre es doch, wenn die Bash »wüßte«, daß *gv* nur mit Dateien aufgerufen werden kann, die die Endung ».ps« besitzen. Mit Hilfe des Shellbefehls *complete* können wir die Bash anlernen:

```
$ complete -G '*.ps' gv
```

Die Option *-G* gibt an, auf welches Suchmuster die Dateien passen müssen, mit denen *gv* aufgerufen werden kann.

complete kann noch viel mehr. Das typische am Unixtool *ssh* ist die Tatsache, daß meistens nur ein paar Rechner zur Wahl stehen. Mit dem folgenden Aufruf können wir der Bash mitteilen, daß *ssh* als erstes Argument immer einen Rechnernamen benötigt, der im lokalen Netz verfügbar ist:

```
$ complete -A hostname ssh
$ ssh [Tab][Tab]
::1         localhost   teroknor      teroknor.ufp        utopia.ufp
```

Damit teilen wir *ssh* eine bestimmte Klasse von Argumenten zu. Weitere sind beispielsweise *file* für Dateinamen oder *user* für Benutzernamen.

Falls Sie sich auch auf Server einloggen möchten, die sich nicht im lokalen Netz befinden, kann uns der vorherige Aufruf natürlich nicht helfen. Die folgende einfache Lösung funktioniert nur, wenn wenige ssh-Accounts zur Wahl stehen:

```
$ complete -W 'quark@utopia.ufp stud17k@mit.edu quark@buch.com'
$ ssh [Tab][Tab]
quark@utopia.ufp      stud17k@mit.edu      quark@buch.com
```

Mit der Option *-W* wird eine Liste mit passenden Wörtern übergeben.

Die hier kurz angeschnittenen Optionen von *complete* können natürlich beliebig kombiniert werden. Außerdem gibt es selbstverständlich noch viele weitere Möglichkeiten, die Vervollständigung zu konfigurieren. Eine detaillierte Beschreibung würde jedoch den Rahmen dieses Kapitels sprengen.

10.7.5 Tastenbelegung ändern

Die Tastenbelegung, wie sie im vorherigen Kapitel besprochen wurde, hat sich im Lauf der Zeit etabliert, doch oft entspricht sie trotzdem nicht den Wünschen des Benutzers. Deshalb gibt es in der Korn-Shell und der Bash die Möglichkeit, die Tastenbelegung beliebig zu verändern. Da es einige unbelegte Zeileneditor-Funktionen gibt, lohnt sich auch dann ein Blick in die Liste aller Zeileneditor-Funktionen, wenn man mit der Standardtastenbelegung zufrieden ist.

Korn-Shell

Zuständig für die Veränderung der Tastenbelegung ist der Shellbefehl *bind*, der im einfachsten Fall in der Korn-Shell folgende Syntax besitzt:

```
bind <Taste>=<Editor-Funktion>
```

Die Liste aller Editor-Funktionen fragt man mit *bind -l* ab oder man wirft einen Blick in die Tabellen der vorherigen Abschnitte. Will man angeben, welche Taste man belegen will, muß man wissen, wie in Unix-Systemen die Funktionstasten abgebildet werden. Am besten machen Sie selbst die Probe auf's Exempel und starten die Bourne-Shell und drücken beispielsweise [Cursor-Auf]:

```
$ sh
$ ^[[A
```

Da die Bourne-Shell keine Funktionstasten belegt, sehen Sie genau die Zeichenkette, die die Cursortaste erzeugt: Hier handelt es sich um ein Escape-Zeichen gefolgt von einer öffnenden eckigen Klammer und einem großem A. Falls Sie nun mehrere Tasten drücken, werden Sie bemerken, daß alle Tasten mit dem Escape-Zeichen beginnen. Aus diesem Grund nennt man solche Zeichenfolgen auch Escapesequenzen.

Um eine Taste mit *bind* zu belegen, muß man glücklicherweise nicht das Escape-Zeichen eingeben. Stattdessen umschreibt man es durch die zwei Zeichen »^« und »[«. Beachten Sie den Unterschied: Die Escape-Taste erzeugt nur *ein* Zeichen, das allerdings durch die Shell durch zwei Zeichen dargestellt wird.

Die Funktionstasten können wir leider nicht direkt mit der Escapesequenz belegen, die wir durch Ausprobieren erhalten haben, da die Shell die Angabe von nur zwei Tasten erlaubt. Genauer gesagt erlaubt sie als erste Taste nur ein sogenanntes Präfix und als zweites eine beliebige Taste. Für die zwei möglichen Präfixe gibt es die Editor-Funktionen *prefix-1* und *prefix-2*. Die Standardbelegung sieht folgendermaßen aus:

```
$ bind | grep prefix
^X = prefix-2
^[ = prefix-1
^[[ = prefix-2
```

Dieses Listing zeigt bereits die Lösung für das Problem der Funktionstasten. Da Editor-Funktionen mehrfach belegt werden können, können wir ein Präfix mit einer einzigen Taste belegen und als Alternative das zweistellige Präfix der Funktionstasten verwenden. Möchten wir nun unsere Cursor-Taste mit der Funktion *up-history* belegen, müssen wir *bind* folgendermaßen aufrufen:

```
$ bind '^XA=up-history'
```

Bash

Die Bash versucht, die umständlichen Klimmzüge der Korn-Shell bei den Funktionstasten zu vermeiden und sieht deshalb eine gänzlich andere Syntax vor. Dies beginnt beim Shellbefehl *bind*:

```
bind <Taste>:<Editor-Funktion>
```

Tastenkombinationen mit der [Strg]-Taste werden mit »\C-x« umschrieben, wobei x für eine beliebige Buchstabentaste steht. Die Meta-Taste, also [Alt] oder [Esc], besitzt die Umschreibung »\e«:

```
$ bind -p | grep backward-char
"\C-b": backward-char
"\eOD": backward-char
"\e[D": backward-char
$ bind -p | grep previous-history
"\C-p": previous-history
"\eOA": previous-history
"\e[A": previous-history
```

Der *bind*-Befehl der Bash bietet eine Fülle von Optionen an:

```
bind [-m <Editormodus>] [-lpsvPSV] [-q <Funktion>]
```

liefert Informationen über die Einstellung des Zeileneditors. Die Optionen:

-m	gibt an, über welchen Editormodus (emacs, emacs-standard, emacs-meta, emacs-ctlx, vi, vi-move, vi-command oder vi-insert) Informationen abgefragt werden sollen
-l	zeigt alle Zeileneditor-Funktionen an
-p	zeigt alle Zeileneditor-Funktionen nebst Tastenbelegung so an, daß die Ausgabe gespeichert und später mit *-f* wieder eingelesen werden kann.
-P	zeigt alle Zeileneditor-Funktionen nebst Tastenbelegung an
-s	zeigt alle mit Zeileneditormakros verbundenen Tastensequenzen so an, daß die Ausgabe gespeichert und wieder eingelesen werden kann.
-S	zeigt alle Tastensequenzen, die mit Zeileneditormakros verbunden sind, sowie die Zeichenketten, die sie ausgeben.
-v	zeigt alle Zeileneditor-Variablen und deren Werte so an, daß die Ausgabe gespeichert und wieder eingelesen werden kann.
-V	zeigt alle Zeileneditor-Variablen und deren Werte an.
-q	zeigt alle Tastenkombinationen an, die mit der angegebenen Zeileneditor-Funktion verknüpft sind.

```
bind [-m <Editormodus>] -f <Dateiname>
```

liest Tastaturbelegungen aus der angegebenen Datei.

```
bind [-m <Editormodus>] -u <Zeileneditor-Funktion>
```

entfernt alle Tastenkombinationen, die mit der angegebenen Zeileneditor-Funktion verbunden sind.

```
bind [-m <Editormodus>] -r <Tastenkombination>
```

befreit die angegebene Tastenkombination von allen Zeileneditor-Funktionen.

```
bind [-m <Editormodus>] -x <Tastenbelegung>:<Kommando>
```

belegt <Taste> so, daß <Kommando> auf Knopfdruck ausgeführt wird.

```
bind [-m <Editormodus>] <Tastenbelegung>:<Zeileneditor-Funktion>
```

belegt die <Taste> mit der angegebenen <Zeileneditor-Funktion>.

10.8 Die Jobverwaltung

Unix kann mit beliebig vielen Prozessen umgehen. Tatsächlich haben Sie nach der Installation von NetBSD vermutlich mindestens zwanzig Prozesse laufen. Um dieser Eigenschaft des Betriebssystems Rechnung zu tragen, besitzt die Shell die Möglichkeit, auch während einer einzigen Shell-Sitzung mehrere Programme zu starten und zu verwalten. So spart man sich den Wechsel zwischen verschiedenen Konsolen oder Terminal-Fenstern – insbesondere natürlich auch bei Hardware-Terminals oder Dial-up-Verbindungen.

10.8.1 Hintergrundprozesse

Möchte man mehrere Programme nebeneinander in einer Shell-Sitzung laufen lassen, stellt sich natürlich ein Problem: Die Programme können sich den Bildschirm (Terminal) nicht teilen. Deshalb muß man dafür sorgen, daß solche Prozesse nur mit Dateien arbeiten, beispielsweise mit Hilfe von Umlenkungen. Da also nur ein Prozeß im Vordergrund ist und alle anderen buchstäblich im Hintergrund arbeiten, nennt man diese Prozesse auch *Hintergrundprozesse*. Man startet sie, indem man an das gewünschte Kommando ein »&« anhängt:

```
$ find . -name \*.o -print > o-files &
[1] 1533
```

Damit der Benutzer den Überblick über alle von ihm gestarteten Prozesse behält, verwaltet die Shell alle Prozesse in der sogenannten Jobverwaltung. Wie hier an diesem Beispiel ersichtlich, gibt die Shell auch sofort preis, welche Jobnummer und welche PID (Process ID Number) dem Hintergrundprozeß zugewiesen wurde. Einen Überblick über alle Jobs der aktuellen Sitzung bietet der Shellbefehl *jobs*:

```
$ jobs
[1]+ Running       find . -name \*.o -print > o-files &
```

Zu unterscheiden ist die Jobnummer von der PID. Die PID wird von Betriebssystem festgelegt und ist dementsprechend auf dem gesamten Rechner gültig und eindeutig. Die Jobnummer wird nur von der Shell vergeben und ist lediglich innerhalb einer Shell-Sitzung gültig.

10.8.2 Status eines Hintergrundprozesses verändern

Vielleicht stellen Sie sich die Frage, was passiert, wenn ein Hintergrundprozeß Eingaben vom Benutzer verlangt und die Standardeingabe nicht umgelenkt wurde. Dieselbe Frage stellt sich natürlich auch bei der Standardausgabe. In diesen Fällen stoppt die Shell den fraglichen Prozeß. Anschließend bleibt dem Anwender die Wahl, entweder die Standardeingabe- beziehungsweise die Standardausgabe umzulenken oder den Prozeß in den Vordergrund zu holen. Betrachten wir zu diesem Thema ein Beispiel:

```
$ find . -name \*.o -print &
$ ...
[1]+ Stopped (tty output)    find . -name \*.o -print &
$
```

find sucht hier solange nach Dateien, bis die Suche das erste Ergebnis liefert. Da Hintergrundprozesse nicht auf die Standardausgabe schreiben dürfen, stoppt die Shell diesen Job. Möchten wir nun die Ergebnisse unmittelbar am Bildschirm betrachten, können wir den Hintergrundprozeß wieder mit *fg* in den Vordergrund holen:

```
$ fg %1
./C/main.o
```

Beachten Sie die spezielle Notation für Jobs, denn vor die Jobnummer muß ein Prozentzeichen gesetzt werden. Die Korn-Shell und die Bash bieten weitere Möglichkeiten, auf Jobs zuzugreifen:

Ausdruck	steht für
%<Jobnummer>	den Job mit der angegebenen Jobnummer
%<Zeichenkette>	den Job, der mit der angegebenen Zeichenkette beginnt. Gibt es keinen solchen Job oder gibt es mehr als einen, gibt die Shell einen Fehler aus.
%?<Zeichenkette>	den Job, der die angegebene Zeichenkette enthält
%+ bzw. %%	den aktuellen Job
%-	den vorherigen Job

Betrachten wir dazu ein paar Beispiele:

```
$ jobs
[1]+  Running              find -name \*.o -print >o-files &
[2]-  Running              bzip2 backup.tar &
[3]   Running              cat < /dev/zero > /dev/null &
$ fg %find
find -name \*.o -print >o-files &
...
$ fg %?zip
bzip2 backup.tar &
...
$ jobs
[3]+  Running              cat < /dev/zero > /dev/null &
$ fg %%
cat < /dev/zero > /dev/null &
```

Kommen wir zu unserem Ausgangsbeispiel zurück. Nehmen wir an, daß unser *find*-Aufruf wahrscheinlich sehr viele Dateien finden wird, und es sich deshalb empfiehlt, die Ergebnisse in eine Datei zu schreiben. In diesem Fall kann der Prozeß natürlich im Hintergrund weiterarbeiten, allerdings muß die Standardausgabe umgelenkt werden. Dies geschieht mit folgendem Aufruf:

```
$ bg %1 > o-files
```

bg sendet einen Prozeß in den Hintergrund. Dasselbe bewirkt auch folgender Aufruf:

```
$ fg %1 > o-files &
```

Einen Fall haben wir noch nicht besprochen, nämlich einen gerade laufenden Prozeß in den Hintergrund zu schicken. Dafür gibt es die Tastenkombination [Strg]+[Z].

```
$ compress verybigfile
^Z
[1]+  Stopped                 compress verybigfile
$ bg %1
[1]+  compress verybigfile
$ ...
[1]+  Done                    compress verybigfile
```

Diese Tastenbelegung kann man mit dem Unixtool *stty* verändern:

```
$ stty susp ^Z
```

»^Z« steht hier ausnahmsweise tatsächlich für das Drücken dieser Taste. Dafür drückt man zuerst [Strg]+[V] und dann [Strg]+[Z].

10.8.3 Prozesse beenden

Prozesse beenden sich normalerweise von selbst. Läuft jedoch ein Programm nicht erwartungsgemäß, steht man vor der Aufgabe, den Prozeß manuell zu beenden. Der Shellbefehl *kill* erledigt diese und viele andere Aufgaben für uns:

```
kill [-<Signal>] <Job oder PID>
```

Auch wenn es der Name andeuten mag, ist *kill* nicht ausschließlich für die Beendigung von Prozessen gedacht, sondern für das Senden von Signalen an Prozessen. Signale kann man sich als sehr kurze Nachrichten vorstellen, die an einen Prozeß beziehungsweise an das verwaltende Betriebssystem gesendet werden. Im einfachsten Fall, also *kill* ohne weitere Optionen, heißt diese Nachricht *TERM*, also Terminierung. Meistens beendet sich der Prozeß dann selbst, hat jedoch noch Gelegenheit, eventuell offene Dateien zu schließen und sauber aufzuräumen:

```
$ ps | grep vi
1417 E0 T    0:00.30 vi main.c
1419 E0 R+   0:00.20 grep vi
$ kill 1417
$ ps | grep vi
1420 E0 R+   0:00.20 grep vi
```

Da es leider auch vorkommen kann, daß Prozesse sich durch das Signal *TERM* nicht mehr beenden lassen, gibt es das schärfere Signal *KILL*. Es wird direkt vom Betriebssystem ausgeführt, so daß dadurch Prozesse garantiert »gekillt« werden können, wie es im Unix-Jargon heißt. Dem Befehl *kill* muß dieses Signal als Option übergeben werden:

```
$ kill -KILL %vi
```

Da die Bourne-Shell keine Signalnamen kennt, müssen wir hier auf die Signalnummer zurückgreifen:

```
$ kill -9 %vi
```

Weitere hilfreiche Signale entnehmen Sie der folgenden Tabelle:

Signalname	Nummer	Beschreibung
HUP	1	Hangup, bewirkt häufig einen Neustart oder ein neues Einlesen der Konfigurationsdateien; wird auch beim Ausloggen an alle Hintergrundprozesse gesendet.
STOP	23	zwingt den empfangenden Prozeß, anzuhalten.
TSTP	24	ist das Suspend-Signal und wird in der Regel durch die Tastenkombination [Strg]+[Z] ausgelöst; daraufhin wird also das Programm angehalten.
CONT	25	mit STOP oder TSTP angehaltene Prozesse wieder aufwekken

10.9 Standardvariablen

In NetBSD ist einzig die C-Shell sinnvoll vorkonfiguriert. Alle anderen Shells kommen mit den Standardeinstellungen daher und sind dementsprechend unhandlich. Dieses und die nächsten Kapitel möchten die Hebel und Schalter aufzeigen, die zu bedienen sind, um aus dem Rohbau ein wohlig warmes Wohnzimmer zu machen.

10.9.1 Standardvariablen in der Bourne-Shell

Standardvariablen, die auch vordefinierte Variablen genannt werden, haben eine besondere Bedeutung: sie konfigurieren die Shell. Einige von dieser Sorte haben wir bereits kennengelernt, beispielsweise *HOME* oder *PATH*. Dieses Kapitel zeigt einige wichtige Standardvariablen, die in der Bourne-Shell verfügbar sind:

CDPATH

CDPATH ist eine spezielle Variable für den *cd*-Befehl, die eine Liste von Verzeichnissen enthält, in denen nach einem Verzeichnis gesucht werden soll. Wird *cd* mit einem relativen Pfad aufgerufen und gibt es dem angegebenen Verzeichnisbaum nicht im aktuellen Verzeichnis, sucht die Shell in den in *CDPATH* angegebenen Verzeichnissen. Alle Verzeichnisse müssen durch Doppelpunkte getrennt werden:

```
$ CDPATH=/:/usr
$ cd tmp
$ pwd
/tmp
$ cd X11R6
$ pwd
/usr/X11R6
$ cd bin              # es wird in der Reihenfolge der Pfade in
$ pwd                 # CDPATH gesucht, deshalb /bin und nicht
/bin                  # /usr/bin!
```

LOGNAME

LOGNAME enthält den Login-Namen des aktuell eingeloggten Benutzers. *$LOGNAME* ist somit identisch mit *`logname`*, aber bedeutend schneller.

MAIL

Sie haben bestimmt schon einmal die Meldung »you have mail.« bekommen, wenn eine neue Mail angekommen ist. Damit die Shell so etwas bemerkt, muß sie wissen, wo sich die Mailbox (Postfach) des Anwenders befindet. Dies steht in *MAIL*. In NetBSD sollte *MAIL* folgenden Wert besitzen:

```
MAIL=/var/mail/$LOGNAME
```

MAILPATH

MAILPATH hat dieselbe Funktion wie *MAIL*, nur daß damit mehrere Mailboxen angegeben werden können. Außerdem läßt sich die Standardmeldung »you have mail.« verändern, indem an die Liste der Mailboxen, die jeweils durch einen Doppelpunkt getrennt werden, mit dem Prozentzeichen die Meldung angehängt wird:

```
MAILPATH=/var/mail/$LOGNAME:/home/quark/Mail/mailings%\
"Die Post war da!"
```

Durch Setzen der Variable *MAILPATH* ist der Wert der Variable *MAIL* unwirksam. In diesem Beispiel würde nun in den Postfächern »/var/mail/$LOGNAME« und »/home/quark/Mail/mailings« nach neuer Post gesucht und bei Ankunft einer Mail die Meldung »Die Post war da!« ausgegeben werden.

MAILCHECK

MAILCHECK legt das Zeitintervall in Sekunden fest, in dem nach neuen Mails in den in *MAILPATH* oder *MAIL* angegebenen Postfächern gesucht wird. Besitzt *MAILCHECK* den Wert 0, prüft die Shell bei jeder Ausgabe des Prompts. Standardwert ist zehn Minuten.

```
MAILCHECK=600
```

PS1

PS1 ist der erste Prompt (Primärprompt), also die Zeichenkette, die an der Eingabeaufforderung zur Eingabe einer Befehlszeile aufruft. Standardmäßig ist *PS1*='$' oder als root *PS1*='# '. Leider ist es bei der Bourne-Shell nicht möglich, dynamische Prompts zu setzen, so daß zum Beispiel stets das aktuelle Verzeichnis angezeigt wird:

```
$ echo $PS1
$
$ PS1="Cui Bono? "
Cui Bono? pwd
/home/quark
Cui Bono? PS1="`pwd` > "
/home/quark > cd ..
/home/quark > pwd
/home
/home/quark > PS1='`pwd` > '
`pwd` > ...
```

PS2

PS2 ist das zweite Prompt (Sekundärprompt), das in allen Situationen angezeigt wird, in denen die Shell noch weitere Eingaben erwartet. Voreinstellung ist der Wert *PS2*='> '.

```
$ echo $PS2
>
$ echo La Le Lu \
> nur der Mann im Mond schaut zu
La Le Lu nur der Mann im Mond schaut zu
```

10.9.2 Standardvariablen der Korn-Shell

Die Korn-Shell erweitert den Fundus der Bourne-Shell um folgende Standardvariablen:

EDITOR

Durch *EDITOR* wird der Editor-Modus der Eingabeaufforderung eingestellt, allerdings nur, wenn *VISUAL* nicht gesetzt ist. Möglich sind die Werte *emacs*, *gmacs* oder *vi*. Ist *EDITOR* nicht gesetzt, ist »die letzte Wahl« */bin/ed*.

ENV

ENV enthält den Pfad zu einem Skript, das bei jedem Start einer interaktiven Korn-Shell (nicht jedoch der Login-Shell) ausgeführt wird. Üblicherweise werden in diesem Skript die Aliase und Funktionen definiert.

FCEDIT

FCEDIT enthält den Pfad zu dem Editor, der durch den Befehl *fc* aufgerufen wird. Voreingestellt ist *FCEDIT="/bin/ed"*.

HISTFILE

gibt an, in welcher Datei die History gespeichert wird. Der Standardwert ist *$HOME/.sh_history*.

HISTSIZE

Mit *HISTSIZE* wird festgelegt, wie viele Befehlszeilen in der History gespeichert werden. Standardwert ist *HISTSIZE=128*.

MAILPATH

Die Handhabe von *MAILPATH* bei der Korn-Shell unterscheidet sich nur in drei Punkten von der Bourne-Shell – zum einen wird das Prozentzeichen durch ein Fragezeichen ersetzt und zum anderen kann in der Meldung, daß neue Mail im Postfach ist, der Name des Postfachs mit »$_« verwendet werden:

```
$ MAILPATH=/var/spool/mail/quark:/home/quark/Mail/mailings?\
> "Du hast neue Post in $_."
$ ...
Du hast neue Post in /home/quark/Mail/mailings.
$ ...
```

Zusätzlich werden in der Meldung dynamisch, also erst bei Ausgabe der Zeichenkette, Kommando- und Variablensubstitution durchgeführt:

```
$ MAILPATH=/var/spool/mail/quark:/home/quark/Mail/mailings?\
> "Hallo $LOGNAME, Du hast neue Post in $_."
$ ...
Hallo quark, Du hast neue Post in /var/spool/mail/quark.
$ ...
```

PS1 und PS2

Ebenso wie bei *MAILPATH* bildet die Korn-Shell auch bei *PS1* und *PS2* die Prompts dynamisch, das heißt eventuell vorkommende Variablen- oder Kommandosubstitutionen werden bei jeder Ausgabe der Zeichenkette neu ausgewertet. So läßt sich ein sehr bequemer Prompt basteln:

```
$ PS1='$LOGNAME@`hostname`:`pwd` ! > '
quark@teroknor:/home/quark 1 > cd ..
quark@teroknor:/home 2 > ...
```

Wie im Beispiel zu sehen ist, steht ein Ausrufezeichen in *PS1* für die Zeilennumer, die die aktuelle Eingabe in der History einnehmen wird.

TMOUT

Der böse Cracker lauert überall. Besonders einfach wird es jedoch auch für Möchte-Gern-Cracker, wenn Sie Ihren Arbeitsplatz ohne sich auszuloggen verlassen. Eine kleinen Schutz vor der eigenen Vergeßlichkeit bietet die Variable *TMOUT*. Ist sie auf einen Wert größer Null eingestellt, loggt Sie die Korn-Shell nach *$TMOUT* Sekunden automatisch aus, falls in diesem Zeitraum kein [Enter] gedrückt wurde.

VISUAL

Mit *VISUAL* wird ebenfalls wie in *EDITOR* den Modus des Zeileneditors eingestellt. *VISUAL* ist jedoch die ausschlaggebende Variable, nur wenn *VISUAL* nicht gesetzt ist, wird der Wert von *EDITOR* verwendet. Mögliche Modi sind *emacs*, *gacs* und *vi*. Standardeinstellung ist *VISUAL="/bin/ed"*, was gleichbedeutend mit »minimale Editierfunktionen« ist.

10.9.3 Standardvariablen der Bash

Die Bash kennt alle Standardvariablen der Korn-Shell und noch viele mehr:

GLOBIGNORE

GLOBIGNORE enthält eine durch Doppelpunkte getrennte Liste von Mustern, wie man sie von der Dateinamenexpandierung kennt. Ist die Variable gesetzt, schließt die Shell alle Dateien, auf die mindestens ein Muster in *GLOBIGNORE* paßt, von der Dateinamenexpandierung aus. Diese Dateien können also nur noch direkt erreicht werden.

```
$ ls -C
main.o   core   main.c   README
$ GLOBIGNORE='core:*.o:*~'
$ echo *
main.c   README
```

HISTIGNORE

Über diese Standardvariable wird festgelegt, welche Zeilen nicht in der History gespeichert werden sollen. *HISTIGNORE* enthält hierzu eine durch Doppelpunkte getrennte Liste von Mustern, wie man sie von der Dateinamenexpandierung kennt. Eine besondere Bedeutung erhält »&«, denn sobald es in *HISTIGNORE* vorkommt, steht es für die zuletzt eingegebene Zeile.

```
$ HISTIGNORE='ls:ls *'
$ history | last -2
  655 HISTIGNORE='ls *:echo *'
  656 history | last -2
$ ls
file1
$ history | last -2
  656 history | last -2
  656 history | last -2
$ HISTIGNORE='&'
```

PROMPT_COMMAND

Der in *PROMPT_COMMAND* enthaltene Befehl wird bei jeder Ausgabe des Promptes ausgeführt.

PS1 und PS2

Mit der Bash sind sehr ausführliche Prompte möglich. Denn zusätzlich zu den dynamischen Variablenausdrücken und Kommandosubstitutionen lassen sich mit bestimmten Sequenzen viele unterschiedliche Informationen im Prompt integrieren:

Sequenz	wird ersetzt durch
\t	die aktuelle Zeit
\d	das aktuelle Datum
\n	einen Zeilenumbruch
\s	den Namen der Shell
\w	das aktuelle Verzeichnis
\W	den Namen des aktuellen Verzeichnisses (bei »/usr/bin« durch »bin«)
\u	den aktuellen Benutzer
\h	den Rechnernamen
\#	die Nummer des aktuellen Befehls
\!	die Nummer der aktuellen Befehlszeile in der History-Datei
\$	#, falls die effektive UID 0 ist (also bei Benutzer root), ansonsten $
\nnn	das Zeichen mit dem ASCII-Code *nnn* (als Oktalzahl angegeben)
\\	den Backslash \

Mit \[lassen sich außerdem ANSI-Escape-Sequenzen einleiten. Das Ende der Sequenz wird durch \] gekennzeichnet. Mit ANSI-Escape-Sequenzen lassen sich Texte auf dem Bildschirm in verschiedenen Farben ausgeben und mit Attributen wie »fett« oder »unterstrichen« versehen. Folgender Prompt druckt das aktuelle Verzeichnis fett:

```
PS1="\u@\h:\[^[[1m\]\w\[^[[0m\] > "
```

^[muß hier mit der Escape-Taste eingegeben werden. In der Shell kann dies mit [Strg]+[V] [Esc] geschehen.

Der Shellexperte, der mit den unterschiedlichsten Shells arbeitet, möchte allerdings gerne wissen, mit welcher Shell und Shellversion er gerade arbeitet:

```
$ PS1='\s \u@\h:\w > '
bash 2.05 quark@teroknor:/home/quark >
```

10.9.4 Standardvariablen der C-Shell

Die C-Shell gehört zu einer ganz anderen Familie von Shells, deshalb müssen wir sie gesondert betrachten. Allerdings kann dies sehr kurz geschehen, da die meisten Variablen in einer ähnlichen Form auch in den anderen Shells vorkommen:

Standardvariable	Beschreibung
cdpath	Suchpfad für cd
filec	aktiviert die Vervollständigung
histfile	History-Datei
history	Größe der History in Zeilen (im Speicher)
ignoreeof	ignoriert [Strg]+[D]
mail	Benutzer-Postfach
noclobber	existierende Dateien bei Umlenkung überschreiben
noglob	schaltet Dateinamenexpandierung aus
nonomatch	falls gesetzt, ist eine fehlerhafte Dateinamenexpandierung kein Fehler
notify	Meldungen der Jobverwaltung asynchron auf den Bildschirm ausgeben, also nicht erst, wenn ein Prompt geschrieben wird
prompt	Das Prompt. »!« wird durch die History-Nummer ersetzt
savehist	Größe der History-Datei

10.10 Shelloptionen

Shelloptionen verändern das Verhalten der Shell. Dieses Kapitel zeigt die wichtigsten Shelloptionen.

10.10.1 Shelloptionen der Bourne-Shell

set ist der Urvater aller Konfigurationsbefehle der Shell. Ihn gab es bereits in der Bourne-Shell, dementsprechend kann er nur sehr grundlegende Verhaltensweisen der Shell verändern. Eine Option setzt man mit

```
set -<Option>
```

Etwas verwirrend ist, daß man eine Option mit

```
set +<Option>
```

wieder löscht. Eigentlich würde man intuitiv annehmen, daß »+« setzt und »-« wieder entfernt. Wenn man sich jedoch vor Augen führt, daß alle Optionen standardmäßig nicht gesetzt sind, macht es Sinn, wenn man die Ausnahme, nämlich das Ausschalten einer Option, auf das unüblichere »+« verlagert.

Die folgende Aufstellung bietet eine Übersicht über einige wichtige Optionen von *set* in der Bourne-Shell:

-f Dateinamenexpandierung ausschalten (filename expansion)

Mit *set -f* kann man die Dateinamenexpandierung ausschalten, falls man mit vielen Dateien arbeitet, deren Namen »*«, »?« oder ähnliche Jokerzeichen enthalten.

```
$ echo *
datei1 datei2 datei3
$ set -f
$ echo *
*
```

-u Zugriff auf undefinierte Variablen verbieten (unset)

Normalerweise ist die Shell sehr großzügig im Umgang mit Variablen. Ist eine Variable nicht gesetzt, wird einfach eine leere Zeichenkette eingesetzt. Mit *set -u* kann diese Funktionalität ausgeschaltet werden.

```
$ unset HALLOGALLO
$ echo $HALLOGALLO

$ set -u
$ echo $HALLOGALLO
sh: HALLOGALLO: unbound variable
```

10.10.2 Shelloptionen der Korn-Shell

Die Korn-Shell kennt neben den Optionen der Bourne-Shell weitere Einstellungsmöglichkeiten. Die folgende Auflistung zeigt alle mit der Korn-Shell neu hinzugekommenen Optionen von *set*:

-m Prozeßverwaltung einschalten (monitor)

Diese Option schaltet die Prozeßverwaltung der Shell ein. Ist sie aktiviert, können die bekannten Shellbefehle *fg*, *bg* oder *jobs* verwendet werden.

```
$ find -name '*.bak' &
[1] 1130
$ jobs
$ bg %1
bg: no job control
```

Um wohl die Gefahr abzuwenden, daß *set* die Buchstaben ausgehen, wurde mit der Korn-Shell eine neue Möglichkeit eingeführt, Optionen zu setzen und zu löschen. Einige Konfigurationsmöglichkeiten können nur über *-o* erreicht werden.

Die folgende Tabelle bietet eine Übersicht über alle gültigen Optionen:

Option	Bedeutung
bgnice	alle Hintergrundprozesse mit niedrigerer Priorität ausführen (entspricht einem Aufruf mit *nice*)
emacs	schaltet den Zeileneditor in den emacs-Modus (alle gängigen emacs-Tastenbelegungen sind aktiviert)
gmacs	schaltet den Zeileneditor in den emacs-Modus (alle gängigen emacs-Tastenbelegungen sind aktiviert). Der einzige Unterschied zu -o *emacs* befindet sich bei der Tastenbelegung von [Strg]+[T]
ignoreeof	eine interaktive Shell kann nicht mit [Strg]+[D], sondern nur mit *exit* beendet werden
markdirs	an alle Verzeichnisnamen, die mit der Dateinamenexpandierung eingesetzt werden, einen Schrägstrich / anhängen
noclobber	Bei der Umlenkung mit > können bereits existierende Dateien nicht überschrieben werden (mit >| ist es weiterhin möglich)
vi	schaltet den Zeileneditor in den vi-Modus (alle gängigen vi-Tastenbelegungen sind aktiviert)
viraw	schaltet den Zeileneditor in den vi-raw-Modus (entspricht dem Eingabemodus von vi)
xtrace	entspricht -x

Zu beachten ist immer, welche Option positiv und welche negativ formuliert ist. Ist beispielsweise *noglob* eingeschaltet, wird die Dateinamenexpandierung ausgeschaltet.

```
$ mkdir verz
$ echo *
datei1 datei2 datei3 verz
$ set -o markdirs
$ echo *
datei1 datei2 datei3 verz/
```

10.10.3 Shelloptionen der Bash

Die Bash kennt alle Optionen des *set*-Befehls der Korn-Shell und bietet darüber hinaus einige neue Konfigurationsmöglichkeiten an:

-b Statusmeldung der Prozeßverwaltung sofort ausgeben

Normalerweise wartet die Shell bei Berichten über die Beendigung von Hintergrundprozessen auf die Ausgabe des nächsten Prompts. Ist diese Option jedoch eingeschaltet, wird der Report unmittelbar direkt auf die Standardausgabe gegeben.

-H History über ! erreichbar machen

Ist diese Option eingeschaltet (Standard), ist die von der C-Shell bekannte History-Substitution auch in der Bash möglich.

```
$ echo Das ganze Leben ist ein Quiz.
$ set -H
$ !e
echo Das ganze Leben ist ein Quiz.
Das ganze Leben ist ein Quiz.
```

-P virtuelles Dateisystem ausschalten (physical)

Diese Option berührt einen Komfort, dessen Sie sich bisher vielleicht noch gar nicht bewußt waren. In NetBSD ist beispielsweise das Verzeichnis »/sys« ein symbolischer Link auf »/usr/src/sys«:

```
$ ls -ld /sys
lrwxr-xr-x  1 root  wheel  11 Sep  6 10:46 /sys -> usr/src/sys
```

Nun wechseln wir in das Verzeichnis »/usr/src/sys«. Daraufhin möchten wir mit *cd ..* in das darunterliegende Verzeichnis gehen. Wo befinden wir uns nun? Was würden Sie erwarten?

```
$ cd /usr/src/sys
$ pwd
/usr/src/sys
$ cd ..
$ pwd
/usr/src
```

Das Ergebnis ist wenig überraschend, doch eigentlich nicht logisch. Denn ein symbolischer Link wird vom Betriebssystem bei Verwendung in einem Systemaufruf in den tatsächlichen Pfad umgesetzt. Sie befinden sich also eigentlich nach *cd /sys* physikalisch im Verzeichnis »/usr/src/sys«. Das überprüft man, wenn man anstatt des Shellbefehls *pwd* das Unixtool einsetzt:

```
$ cd /sys
$ /bin/pwd
/usr/src/sys
```

Die Shell gaukelt uns also etwas vor! Sie verbirgt die tatsächliche Verzeichnisstruktur und verhüllt sie mit einer logischen Verzeichnisstruktur. Dieses Verhalten kommt der Intuition und der Praxis wesentlich näher als das tatsächliche Verhalten. Trotzdem läßt sich der Mechanismus durch *set -P* ausschalten. Dasselbe erreicht man übrigens, wenn man *cd* mit der Option *-P* verwendet.

Neben dem Shellbefehl *set* gibt es in der Bash auch noch der Shellbefehl *shopt*. Optionen setzt man mit

```
shopt -s <Optionen>
```

Dementsprechend löscht man Optionen mit

```
shopt -u <Optionen>
```

Standardmäßig löscht *shopt* die übergebenen Optionen. Folgende Auflistung zeigt die wichtigsten Optionen:

```
cdable_vars
```

Bei eingeschaltetem *cdable_vars* interpretiert die Shell das Argument eines *cd*-Befehls als Variable, falls der Befehl kein Verzeichnis mit dem angegebenen Namen gefunden hat:

```
$ shopt -s cdable_vars
$ TMP=/tmp
$ cd TMP
$ pwd
/tmp
```

cdspell

erlaubt geringfügige Fehler bei Verzeichniswechsel mit *cd*:

```
$ shopt -s cdspell
$ mkdir DDR
$ cd GDR
DDR
$ pwd
/home/quark/DDR
```

checkwinsize

Ist diese Option gesetzt, überprüft die Bash nach jedem Aufruf eines Befehls die Dimensionen des Terminals und setzt gegebenenfalls die Standardvariablen *LINES* und

COLUMNS auf die neuen Werte. Es ist unbedingt zu empfehlen, daß diese Option gesetzt ist. Sonst kann es innerhalb von xterm-Fenstern zu unvorhersehbaren Problemen mit dem Zeileneditor kommen.

dotglob

Wir wissen bereits, daß bei der Dateinamenexpandierung standardmäßig alle versteckten Dateien (Dateien, deren Name mit einem Punkt beginnt) nur dann gefunden werden, wenn vor das Suchmuster explizit ein Punkt gestellt wird:

```
$ touch .morning_bell morning_bell
$ echo *m*
morning_bell
$ echo .*m*
.morning_bell
$ echo .#*m*
.morning_bell morning_bell
```

Setzt man die Option *dotglob*, werden auch versteckte Dateien standardmäßig in das Suchergebnis mit aufgenommen:

```
$ shopt -s dotglob
$ echo *m*
.morning_bell morning_bell
```

histappend

Standardmäßig überschreibt die Bash die History-Datei »$HISTFILE«. Ist diese Option gesetzt, hängt sie die neuen Zeilen an das Ende der alten Datei an.

histreedit

Benutzt man in der Bash eine History-Substitution und ist sie fehlerhaft, hat man normalerweise keine Möglichkeit, das zuvor eingegebene zu verbessern, da History-Substitutionen nicht in die History aufgenommen werden. Ist diese Option jedoch gesetzt, schreibt die Bash die fehlgeschlagene History-Substitution nochmals in den Zeilenpuffer, um sie korrigieren zu können.

histverify

Diese Option ermöglicht die konsequente Verbindung zwischen der History-Substitution und dem Zeileneditor. Ist sie gesetzt, wird das Ergebnis einer History-Substitution nicht unmittelbar von der Shell ausgeführt, sondern in den Zeilenpuffer geschrieben, so daß der Anwender noch manuelle Änderungen im Zeileneditor machen kann.

hostcomplete

schaltet die Vervollständigung von Rechnernamen ein. Dies passiert immer, wenn ein Wort das Zeichen »@« enthält.

huponexit

Ist *huponexit* gesetzt, wird beim Ausloggen jedem Hintergrundprozeß das Signal *HUP* gesendet. Dies ist notwendig, falls man sich via *ssh* eingeloggt hat. *ssh* beendet sich nämlich erst dann, wenn alle in der Login-Sitzung gestarteten Prozesse auf dem entfernten Rechner beendet wurden. Möchte man jedoch einen Prozeß auch nach dem Ausloggen weiterlaufen lassen, muß das Signal *HUP* gesendet werden. Dieser Trick funktioniert in den meisten Fällen.

no_empty_cmd_completion

vermeidet, daß nach Programmen in *PATH* gesucht wird, wenn die Vervollständigung in einer leeren Zeile aktiviert wird. Dies soll verhindern, daß ein versehentliches Drükken der Tabulator-Taste eine aufwendige Suche auslöst.

nocaseglob

schaltet die Unterscheidung zwischen Groß- und Kleinschreibung bei der Dateinamenexpandierung aus:

```
$ ls
anna    ANNA
$ ls *n*
anna
$ shopt -s nocaseglob
$ ls *n*
anna    ANNA
```

nullglob

Standardmäßig verbleiben die Suchmuster der Dateinamenexpandierung unaufgelöst, falls keine passende Datei gefunden wurde:

```
$ echo wrzlbrmpft*
wrzlbrmpft*
```

Ist allerdings *nullglob* gesetzt, werden erfolglose Suchmuster durch die leere Zeichenkette (Länge 0) ersetzt:

```
$ shopt -s nullglob
$ echo wrzlbrmpft*
$ ...
```

Beachten Sie, daß es sehr hilfreich sein kann, wenn diese Option *nicht* gesetzt ist, da dann beispielsweise bei einem *find*-Aufruf oft kein Quoting benötigt wird:

```
$ find . -name *file_name* -print
```

promptvars

schaltet die Expandierung von Variablen im Prompt ein.

10.10.4 Shelloptionen der C-Shell

Die C-Shell besitzt keine Shelloptionen im engeren Sinn. Deren Aufgabe übernehmen hier einige Standardvariablen, die bereits in Kapitel 10.9.4 beschrieben wurden.

10.11 Konfigurationsdateien

In den letzten Kapiteln haben wir einige Möglichkeiten kennengelernt, die Shell zu konfigurieren. Hier beschäftigen wir uns nun damit, wie man diese Einstellungen dauerhaft macht.

10.11.1 Einführung

Beim Stichwort *Konfigurationsdatei* denken viele sofort an die typischen Windows-Dateien, die die Endung ».INI« besitzen. In der Shell gibt es – bis auf eine Ausnahme – keine solche Datei. Vielmehr besteht eine Konfigurationsdatei einfach aus einer Aneinanderreihung von Shell-Kommandos – ein Shellskript. Diese spezielle Art von Shellskript nennt man Startskript, da es die Kommandos enthält, die als erstes ausgeführt werden.

Nun stellt sich natürlich die Frage, *wann* ein solches Startskript ausgeführt wird. Die Antwort ist gar nicht so einfach, wie es scheint, da die verschiedenen Shells unterschiedliche Vorstellungen in dieser Sache haben. Grundlegend können wir für unsere vier Shells folgende Arten von Startskripten unterscheiden:

- Das Login-Skript wird nur dann ausgeführt, wenn die Shell als Login-Shell gestartet wurde.
- Das Interactive-Session-Skript wird nur dann ausgeführt, wenn die Shell nicht als Login-Shell und nicht zur Ausführung eines Skripts ausgeführt wird. Beispielsweise könnte dies durch Aufruf an der Kommandozeile geschehen.
- Das Non-Interactive-Skript wird nur dann ausgeführt, wenn die Shell zur Ausführung eines Shellskripts aufgerufen wird.
- Das Session-Skript wird bei jedem Aufruf der Shell ausgeführt.

Weiterhin unterscheiden wir zwischen globalen, also systemweit gültigen, und benutzerspezifischen Startskripten. Globale Skripte sind im Verzeichnis »/etc« angesiedelt. Falls beide Skripte vorhanden sind, muß man zusätzlich beachten, ob in diesem Fall beide oder nur eines von beiden ausgeführt wird.

Sind Sie nun verwirrt? Ganz so schwierig, wie es hier dargestellt ist, ist es nicht, wenn man sich auf eine einzige Shell beschränkt. Die folgenden Kapitel zeigen das Verhalten der vier besprochenen Shells. Falls Sie jedoch mehrere Shells benutzen möchten oder Systemadministrator für ein Mehrbenutzersystem sind, sollten Sie sich einmal Gedanken darüber machen. Im Großen und Ganzen ist es sinnvoll, eine gewisse Systematik bei der Verwaltung von Startskripten einzuhalten. Beispielsweise können die Alias-Definitionen der Korn-Shell und der Bash in einer eigenen Datei geschehen, da diese dieselbe Syntax besitzen.

10.11.2 Die Konfigurationsdateien der Bourne-Shell

Die Bourne-Shell besitzt zwei Login-Skripte: Die globale Datei »/etc/profile« und die benutzerspezifische Datei »$HOME/.profile«. Gibt es beide Dateien, wird zuerst das globale und dann das benutzerspezifische Startskript ausgeführt. In NetBSD sind beide Dateien bereits vorhanden und mit sinnvollen Einstellungen versehen, so daß hier nur noch höchstens kleinere Detailveränderungen notwendig sind.

Skript	Typ
/etc/profile	Loginskript
$HOME/.profile	Loginskript

Die Skripte werden in der in der Tabelle angegebenen Reihenfolge ausgeführt.

10.11.3 Die Konfigurationsdateien der C-Shell

Die C-Shell ist eindeutig die favorisierte Shell von NetBSD, da sie am besten konfiguriert ist. Sie unterstützt neben einem Login-Skript auch ein Logout-Skript, das nach dem Ausloggen ausgeführt wird:

Skript	Typ
/etc/csh.cshrc	Session-Skript
~/.cshrc	Session-Skript
/etc/csh.login	Login-Skript
~/.login	Login-Skript
/etc/csh.logout	Logout-Skript
~/.logout	Logout-Skript

10.11.4 Die Konfigurationsdateien der Korn-Shell

Während Bourne-Shell und C-Shell gut konfiguriert sind, kommt die Korn-Shell in NetBSD als Stiefkind in einer Rohform daher. Unser Ziel, eine gute Konfiguration für die Korn-Shell zusammenzustellen, wird von einem Kompatibilitätsproblem überschattet: Bourne-Shell und Korn-Shell müssen sich dieselben Startskripte teilen:

Skript	Typ
/etc/profile	Login-Skript
$HOME/.profile	Login-Skript
$ENV	Interactive-Session-Skript

Um die Skripte möglichst übersichtlich zu halten, empfehle ich, trotzdem ein eigenes Startskript für die Korn-Shell zu verwenden und dieses dann in »*profile« einzulesen. Wie dies funktioniert, zeigen die folgenden Listings:

```
# Aliase für ksh und bash
alias j='jobs -l'
alias la='ls -a'
alias lf='ls -FA'
alias ll='ls -lA'
alias l='ls -CF'
```

Listing 10.1: $HOME/.aliases

```
#       $NetBSD: dot.profile,v 1.1 2000/10/16 13:12:26 simonb Exp $
PATH=$HOME/bin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin:/usr/pkg/bin
PATH=${PATH}:/usr/pkg/sbin:/usr/games:/usr/local/bin:/usr/local/sbin
export PATH

EDITOR=vi
export EDITOR
EXINIT='set autoindent'
export EXINIT
PAGER=more
export PAGER

umask 2
```

```
# Korn-Shell-Profile aufrufen
if test -n "$KSH_VERSION"; then
   . $HOME/.ksh_profile
fi
```

Listing 10.2: $HOME/.profile

```
# Zeileneditor einschalten
set -o emacs

# Aliase einbinden
. $HOME/.aliases

# Standardvariablen setzen
export HOSTNAME=$(hostname) LOGNAME=$(whoami)
export FCEDIT=/usr/bin/vi HISTFILE=$HOME/.ksh_history HISTSIZE=1000
export MAIL=/var/mail/$LOGNAME MAILCHECK=600
export PS1="$LOGNAME@$HOSTNAME:\$PWD > "

#Tabulator-Taste zur Vervollständigung
/bind '^I'=complete '^I'=complete-list
```

Listing 10.3: ~/.ksh_profile

10.11.5 Die Konfigurationsdateien der Bash

Die Bash bietet die größte Auswahl an Startskripten:

Skript	Typ
/etc/profile	Login-Skript
~/.bash_profile	Login-Skript [1]
~/.bash_login	Login-Skript [1]
~/.profile	Login-Skript [1]
~/.bashrc	Interactive-Skript
$BASH_ENV	Session-Skript

[1] Die Bash sucht eines von diesen angegebenen Skripten in der angegebenen Reihenfolge und führt nur das erste gefundene aus.

Die Trennung zwischen Login-Shell und Interactive-Shell sollte man folgendermaßen umgehen:

```
if [ -f ~/.bashrc ]; then . ~/.bashrc; fi
```

Listing 10.4: ~/.bash_profile

Hiermit wird in der Datei »~/.bash_profile« die Datei »~/.bashrc« ausgeführt.

Ebenso wie die Korn-Shell ist die Bash in NetBSD nicht konfiguriert, weshalb wir die Konfigurationsdatei selbst erstellen müssen:

```
# Aliase einbinden
. $HOME/.aliases

# .profile der Bourne-Shell einlesen
. $HOME/.profile

# Standardvariablen setzen
export FCEDIT=/usr/bin/vi HISTSIZE=1000
export MAIL=/var/mail/$LOGNAME MAILCHECK=600
export PS1='\u@\h:\w > '

# praktische Shelloptionen
shopt -s cdspell histappend huponexit
```

Listing 10.5: ~/.bashrc

Beachten Sie auch, daß bei deutscher Tastaturbelegung unbedingt die Datei »/etc/inputrc« wie in Kapitel 10.7.1 beschrieben konfiguriert werden muß.

10.12 Ausblick

Sie können nun die Kommandozeile der Shell bedienen und haben bereits erfolgreich die Umgebung an Ihre Bedürfnisse angepaßt. War das schon alles? Die Antwort lautet ganz klar nein: Eigentlich kann die wahre Mächtigkeit der Shell erst in Shellskripten zur Geltung kommen. Auf diese Thematik sind wir bisher nur sehr kurz eingegangen. In diesem Buch ist kein Platz für die Shellprogrammierung, schließlich gibt es dicke Bücher darüber. Falls Sie sich für das Thema Shellprogrammierung interessieren, können Sie einen Blick in mein Buch »Shell-Programmierung in Unix« werfen, ebenfalls im C&L-Verlag erschienen.

11 Befehle der Kommandozeile

von Norbert Braun

Unix ist berühmt und berüchtigt für die Vielfalt und Mächtigkeit seiner Kommandos. Lassen Sie sich davon aber nicht abschrecken. Im wesentlichen reicht ein kleiner Teil dieser Befehle aus, um zurechtzukommen. Zudem sind viele Befehle nur für Benutzer mit Administrator-Rechten, also für den »Superuser«, interessant. Das folgende Kapitel soll einen Überblick über die interessantesten Kommandos geben, kann aber nicht vollständig sein, da sonst einige hundert Seiten zusammengekommen wären.

Dieses Kapitel ist in drei Bereiche eingeteilt: Zunächst werden anhand einer Einführung in Unix Anweisungen vorgestellt, die für »normale« Benutzer sinnvoll einsetzbar sind. Dann folgen einige wichtige Befehle, die hauptsächlich für Superuser von Interesse sind und zum Schluß, sozusagen als Bonbon, erfolgt noch eine Einführung in spezielle Unix-Tools wie »awk« und »vi«.

11.1 Allgemeine Befehle

11.1.1 Setzen des persönlichen Paßworts

passwd [Username]

Oftmals vergeben Administratoren die Paßwörter zunächst als einfache Floskel, wie beispielsweise *anfang*, *beginn* oder das Geburtsdatum des Benutzers. Die allererste Handlung eines Unix-Anwenders sollte daher darin bestehen, das Paßwort neu zu definieren. Dies geschieht mit dem Kommando *passwd*. Falls bereits vorhanden, fragt das System nach dem alten Paßwort und dann zweimal nach dem neuen, um Tippfehler zu vermeiden. Zwischen Groß- und Kleinschreibung wird unterschieden. Im allgemeinen sind möglichst komplizierte Konstruktionen erwünscht. So sollte das Paßwort niemals zu kurz sein, aus Groß- und Kleinbuchstaben sowie Zahlen bestehen. Obwohl Systemadministratoren im allgemeinen auch Zeichen wie Plus »+«, Minus »-« und Unterstrich »_« gerne sehen, sollten Sie sich das gut überlegen. Es soll in manchen Netzwerken schon vorgekommen sein, daß die Tastaturbelegung an Terminals erst nach dem Einloggen auf deutsch gesetzt wurde – und wo war denn nun noch einmal der verflixte Unterstrich auf der amerikanischen Tastatur? Falls alle Stricke reißen, hat der Administrator immer die Möglichkeit, das Paßwort eines Users neu zu setzen. Vergißt aber der Administrator sein Paßwort, hat er ein Problem.

11.1.2 Erste Hilfe

Eine der wichtigsten Hilfen unter Unix sind die Online-Manual-Pages. In diesem »Handbuch auf der Kommandozeile« finden Sie ausführliche Informationen über die vorhandenen Befehle. Leider sind diese Informationen oftmals derartig ausführlich, daß die wichtigen Parameter darin untergehen. Sie werden in dieser Einführung auch nicht alle Parameter zu jedem Befehl finden. Dafür gibt es ja die Manual-Pages.

man [-k] name

Parameter:

-k	Keyword, man sucht in den Seiten nach dem Wort und gibt eine Liste mit entsprechenden Manual-Pages aus.

Als erste Übung können Sie sich einmal die Hilfe zur Hilfe selbst ansehen. Geben Sie also folgendes ein:

```
man man
```

Nun erhalten Sie alle Parameter des Kommandos *man*.

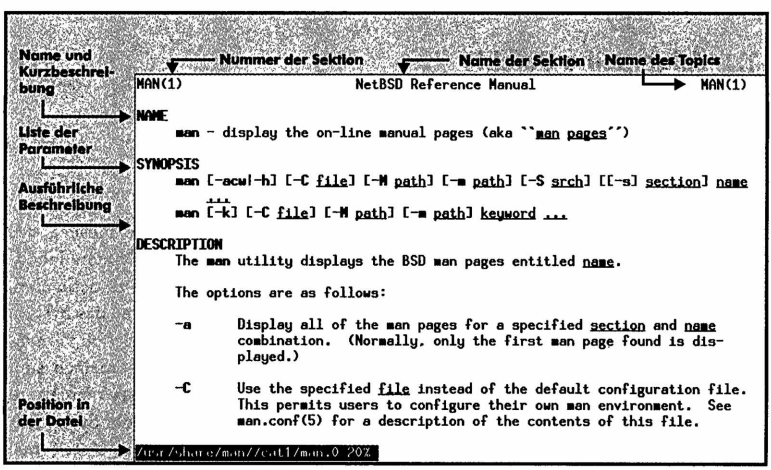


Abb. 11.1: Die erste Seite des Aufrufs von man man

Die Manpages sind in Sektionen aufgeteilt. In Beschreibungen ist oft die Nummer der Sektion, in denen sich die Beschreibung des Befehls befindet, in Klammern hier dem Namen des Befehls angegeben, beispielsweise mit *man(1)*.

Der Aufruf der entsprechenden Sektion findet dann vor dem Namen des Befehls statt:

```
man 1 man
```

Interessant ist das aber nur, wenn es zu einem Befehl mehrere Manpages in verschiedenen Sektion gibt, beispielsweise im Bereich C-Programmierbefehle und Anwendungsprogramme. Die Sektionen sind:

1	Allgemeine Befehle und Anwendungsprogramme
2	Systemaufrufe
3	Bibliotheksfunktionen
4	Kernel-Interface
5	Dateiformate
6	BSD-Spiele
7	Diverses
8	System-Management
9	Kernel-Developer-Handbuch

Bildschirm löschen

clear

Mit dem Aufruf von *clear* wird der Inhalt des aktuellen Bildschirms gelöscht. Der Aufruf entspricht dem DOS-Befehl »cls«.

Sortieren von Daten

sort

Der Befehl *sort* kennt zahlreiche Parameter. Normalerweise wird er aber in Verbindung mit Pipelines eingesetzt, um Texte zu sortieren. Die folgende Zeile beispielsweise gibt den Inhalt eines Verzeichnisses sortiert und bildschirmweise aus:

```
ls -la | sort | more
```

Beachten Sie, daß bei diesem Aufruf nicht nach den Dateinamen, sondern nach den Dateiattributen sortiert wird!

11.2 Leben im Verzeichnisbaum

Um sich durch die Verzeichnisse bewegen zu können, müssen Sie erst ein wenig über die Namenskonventionen von Unix erfahren.

Datei- und Verzeichnisnamen können beliebig viele Punkte ».« enthalten und auch die berühmte 8.3-Längenbegrenzung ist Unix unbekannt. Dies zeigt sich zum Beispiel bei Dateien, die mit *tar* archiviert und mit *gzip* gepackt wurden. Diese Dateien besitzen häufig die Endung ».tar.gz«. So sind Dateinamen wie zum Beispiel »meine_Datei_vom_10.10.96.txt« möglich. Ein Punkt am Anfang des Dateinamens wirkt als »hidden« Attribut. Eine besondere Stellung kommt auch der Tilde »~« zu. Sie dient als Platzhalter für Ihr Home-Directory. Dann ist auch noch zu beachten, daß Unix stets

zwischen Groß- und Kleinschreibung unterscheidet. In Unix werden ähnliche Wildcards verwendet wie unter MS-DOS. Der Stern »*« dient als Platzhalter für eine beliebige Buchstabenkette. Grundsätzlich werden diese Wildcards vom Kommandointerpreter ausgewertet, der dem zu startenden Programm wieder alle Dateien einzeln, mit Leerzeichen getrennt, in die Parameterliste schreibt.

Der Verzeichnisbaum unter Unix beginnt immer in der Root. Der Name dieses Verzeichnisses ist ein einfacher Schrägstrich »/«. Mit diesem Schrägstrich werden auch Verzeichnisnamen getrennt. Im Gegensatz zu MS-DOS, in dem Laufwerke mit Buchstaben symbolisiert werden und jeweils einen eigenen Verzeichnisbaum besitzen, gibt es unter Unix nur ein einziges Verzeichnissystem. Zusätzliche Laufwerke werden jeweils als Unterverzeichnis eingeblendet. Dieser Vorgang nennt sich »mounten«. Eingebunden werden können einfache, lokale Datenträger wie CD- oder Diskettenlaufwerke, aber auch externe Geräte über das Netzwerk. Oftmals ist beispielsweise das Verzeichnis »/home«, in dem die Benutzerdaten gespeichert werden, auf einem bestimmten Server zentralisiert. Es wird dann von den Workstations beim Hochfahren des Systems eingebunden. Da persönliche Einstellungen in Textdateien im Homeverzeichnis des jeweiligen Benutzers abgelegt werden, merkt der Benutzer keinen Unterschied beim Einloggen zwischen den jeweiligen Workstations. Wenn ein solcher Datenträger nun gemountet wurde, darf er keinesfalls einfach aus dem Laufwerk genommen werden. Das Dateisystem des Datenträgers kann sonst beschädigt werden! Um einen Datenträger zu entfernen, *muß* er vorher wieder mit *umount* aus dem Dateisystem ausgebunden werden.

Nach dem Einloggen befindet sich der Unix-Anwender zunächst in seinem persönlichen Stammverzeichnis (Home Directory). In diesem Verzeichnis ist er König. Er kann Dateien und Verzeichnisse anlegen, löschen und deren Zugriffsrechte ändern. Das Home Directory befindet sich üblicherweise unter dem Verzeichnis »/home« und hat den Login-Namen des Users. Den Inhalt eines Verzeichnisses erhält man mit dem Kommando *ls*.

11.2.1 Anderes Rootverzeichnis setzen

chroot [neue_Root] Befehle

Der chroot-Befehl wird verwandt, um ein beliebiges Verzeichnis zum Hauptverzeichnis zu machen. Der Zugriff auf das Original-Rootverzeichnis und dessen Unterverzeichnisse ist damit nicht mehr möglich.

Dieser Befehl findet sowohl bei Servern statt, die dem einloggenden Benutzer nur den Zugriff auf einen bestimmten Teil des Verzeichnisbaums erlauben, wie auch im Linux-Emulator, der automatisch ein *chroot* auf die Linuxverzeichnisse vornimmt. Die Funktion dieses Befehls zeigt das folgende Beispiel:

```
root@beastie: export SHELL=/bin/bash
root@beastie: ~#chroot /compat/linux
I have no name!@beastie: /#ls
bin  boot  etc  lib  mnt  opt  proc  sbin  usr  var
I have no name!@beastie: /#exit
```

```
exit
root@beastie: ~#
```

Es ist wichtig, daß die entsprechend definierte Shell in einem richtigen Unterverzeichnisse des neuen Root-Verzeichnisses ist. Falls dem nicht so ist, muß die Umgebungsvariable *SHELL* entsprechend geändert werden. Der Befehl wird nämlich nur durchgeführt, wenn die entsprechende Shell gefunden wird.

Wenn für bestimmte Benutzer ein eigenes Root-Verzeichnis gesetzt werden soll, müssen dort einige Systembefehle in »/bin« und ein Verzeichnis »/etc« mit den wichtigsten Konfigurationsdateien vorhanden sein. Genaueres hierzu können Sie im Kapitel zum FTP-Daemon nachlesen.

chroot muß nicht parameterlos ausgeführt werden. Es ist genauso möglich, nur einen einzelnen Befehl oder ein Shellskript unter Angabe des alternativen Root-Verzeichnisses zu starten. Nach der Ausführung wird zurückgekehrt:

```
chroot /Verzeichnis auszuführender_Befehl
```

Zu beachten ist, daß natürlich auch im gewechselten Verzeichnis auf den auszuführenden Befehl zugegriffen werden kann.

11.2.2 Verzeichnis anzeigen

***ls** [-a -d -F -i -L -l -R -s -C -x] [Datei...][Verzeichnis...]*

Gibt den Inhalt eines Verzeichnisses aus oder liefert nur Informationen zu den spezifizierten Dateien.

Die Standardausgabe von *ls* auf den Bildschirm ähnelt dem *dir /w* von MS-DOS. Die Dateinamen werden in mehreren Spalten auf dem Terminalfenster ausgegeben. Falls die Ausgabe jedoch in eine Datei umgeleitet wird, wird jeder Dateiname automatisch in eine eigene Zeile geschrieben.

Parameter:

-a	Alle Dateien werden aufgeführt, auch die, die mit einem ».« beginnen.
-d	Ist eine Datei ein Verzeichnis, wird der Name, aber nicht das Inhaltsverzeichnis ausgegeben.
-F	Verzeichnisse werden mit einem »/« markiert.
-i	Gibt zusätzlich die Nummern der belegten I-Nodes aus.
-L	Die Ziele der symbolischen Links werden angezeigt anstatt der Links selbst.
-R	Rekursive Ausgabe auch der Unterverzeichnisse.

-k	Gibt die Dateigröße in Kbyte aus.
-m	Gibt die Dateigröße in Mbyte aus.
-s	Gibt die Dateigröße in Blöcken zu 512 Byte aus.
-C	Es wird eine mehrspaltige Ausgabe erzeugt.
-l	Es wird das ausführliche Format mit Zugriffsrechten, Anzahl der Verweise, Name des Eigentümers, Name der Gruppe, Größe der Datei in Byte, Änderungsdatum und der Dateityp ausgegeben. Für den Dateityp stehen folgende Bezeichnungen: - Normale Datei *d* Verzeichnis *b* Blockorientierte Geräte *c* Zeichenorientierte Geräte *p* FIFO-Puffer oder Named Pipes Für die Zugriffsrechte Eigentümer, Gruppe und den Rest der Welt gibt es die folgenden Bezeichnungen: *r* Leserecht *w* Schreibrecht *x* Ausführungsrecht s SUID (Set User ID) Bit: Versieht den Prozeß bei Ausführung der Datei auf den Besitzer der Datei. Dieses Bit stellt eine mögliche Sicherheitslücke dar. Die Verwendung kann mit einem Kernelschalter gesperrt werden. - Falls kein Recht gesetzt ist

Beispiel:

```
ls -lisa datei1 .profile verzeichnis1
```

Zeigt die Dateien »datei1«, ».profile« und den Inhalt von »verzeichnis1« ausführlich an.

```
ls -al
```

Zeigt alle Dateien, inklusive versteckten, im aktuellen Verzeichnis als ausführliche Liste an.

Falls eine seitenweise Anzeige gewünscht wird, kann dies durch eine Pipe-Umleitung zum Befehl *more* geschehen.

Beispiel:

```
ls -al | more
```

11.2.3 Verzeichnis wechseln

Zum Wechseln des Verzeichnisses wird der Befehl *cd* verwendet.

cd [Verzeichnis]

Beispiele:

cd oder auch *cd ~* wechselt in das Home-Verzeichnis des Anwenders.

cd .. wechselt in das übergeordnete Verzeichnis des gerade aktuellen Verzeichnisses.

cd unterverzeichnis1 wechselt in »unterverzeichnis1« des gerade aktuellen Verzeichnisses.

cd /sub/sub/verzeichnis1 wechselt direkt in »verzeichnis1« im Pfad »/sub/sub«.

cd ~user1 wechselt in das Home-Verzeichnis des Anwenders »user1«.

Beachten Sie, daß im Gegensatz zu DOS/Windows und OS/2 zwischen dem »cd« und den beiden Punkten beim Wechsel zu einer Ebene tiefer und auch beim *cd /* immer ein Leerzeichen zwischen Befehl und Parameter stehen muß. Ein

```
cd..
```

führt zur Meldung

```
bash: cd..: command not found
```

wobei dies kein Problem der Kommandozeilenshell »bash« ist.

Falls Sie als DOS/Windows-Umsteiger auf das »cd..« nicht verzichten wollen, können Sie sich in der lokalen Konfigurationsdatei einen *alias* dafür und für die anderen leerzeichenlosen Befehle definieren:

```
alias cd..='cd ..'
alias cd...='cd ../..'
alias cd/='cd /'
```

Bei der *bash* wären die Eintragungen in der Datei »~/.profile« zu tätigen.

11.2.4 Freien Laufwerksplatz anzeigen

df [-k | -m | -P] [-ailn] [-t Typ] [Datei | Dateisystem ...]

df zeigt die Statistik über den freien Platz auf dem angegebenen Dateisystem (oder ohne dessen Angabe auf allen eingebunden Datenträgern) an. In der Voreinstellung wird die Größe in Blöcken à 512 Byte angezeigt.

df kennt die folgenden Optionen:

-a	Anzeigen der Daten aller Mountpunkte, auch solche, die mit dem Flag *MNT_IGNORE* eingebunden sind.
-i	Einbinden der Statistik der freien Inodes.
-k	Für die Anzeige werden 1-Kbyte-Blöcke verwendet. Diese Option überschreibt die Umgebungsvariable *BLOCKSIZE*.
-l	Es werden nur Informationen zu lokal eingebundene Dateisysteme gezeigt.
-m	Für die Anzeige werden 1048576-Byte (1-Mbyte-Blöcke) verwendet. Diese Option überschreibt die Umgebungsvariable *BLOCKSIZE*.
-n	Ausgeben vorhandener Informationen zu den Dateisystemen. Dateisysteme, die derzeit keine Informationen liefern können, werden ignoriert.
-P	Grundsätzlich soll die POSIX-kompatible Größenangabe in 512-Byte-Blöcken erfolgen, auch wenn die Variable *BLOCKSIZE* im Environment einen anderen Wert enthält.
-t	Nur die Angaben zu dem in *-t* spezifizierten Dateisystem sollen angeben werden. Mehrere Typen werden durch Kommata voneinander getrennt.

Hier einige Beispiele:

```
root@beastie: /#df -m
Filesystem  1K-blocks     Used     Avail Capacity  Mounted on
/dev/wd1a      211526    81812    119137    40%    /
/dev/wd1e     4293230  3454630    623938    84%    /usr
/dev/wd2e     4768228  4183978    345838    92%    /pub
kernfs              1        1         0   100%    /kern
procfs              4        4         0   100%    /proc
root@beastie: /#df -mtffs
Filesystem  1K-blocks     Used     Avail Capacity  Mounted on
/dev/wd1a      211526    81812    119137    40%    /
/dev/wd1e     4293230  3454630    623938    84%    /usr
/dev/wd2e     4768228  4183978    345838    92%    /pub
root@beastie: /#
```

11.2.5 Belegten Speicherplatz ermitteln

***du** [-H | -L | -P] [-a | -s] [-c] [-k] [-x] [file ...]*

Mit dem Befehl *du* wird die aktuelle Belegung eines Dateisystems angezeigt. Wenn keine Datei angegeben ist, wird die Hierarchie des aktuellen Verzeichnisses ausgewertet.

du kennt die folgenden Optionen:

-H	Es wird symbolischen Links auf der Kommandozeile, aber nicht symbolischen Links in der Dateihierarchie gefolgt.
-L	Symbolischen Links auf der Kommandozeile und in der Dateihierarchie wird gefolgt.
-P	Symbolischen Links wird nicht gefolgt (Voreinstellung).
-a	Für jede Datei in der Hierarchie wird ein Eintrag angezeigt.
-c	Zusammenfassung anzeigen.
-k	Anzeige der Größe in 1-Kbyte-Blöcken.
-m	Anzeige der Größe in 1-MByte-Blöcken.
-r	Warnungen zu nicht lesbaren Verzeichnissen anzeigen.
-s	Nur die Gesamtgröße der Dateien anzeigen.
-x	Dateisystem-Mountpunkte werden nicht übersprungen.

11.2.6 Verzeichnisse vergleichen

diff [-r -f] Verzeichnis1 Verzeichnis2

Vergleicht zwei Verzeichnisse miteinander und gibt den Unterschied aus.

Parameter:

-r	Führt das Kommando rekursiv in den Unterverzeichnissen aus.
-s	Zeigt identische Dateien an.

11.2.7 Verzeichnis erstellen

mkdir [-m Mode] [-p] Verzeichnis...

Erstellt ein Verzeichnis.

Parameter:

-m Mode	Definiert wie beim Kommando *chmod* die Zugriffsrechte des neuen Verzeichnisses.
-p	Erzeuge fehlende Zwischenverzeichnisse. Damit wird ein kompletter Pfad mit allen angegebenen Verzeichnissen erzeugt.

Beispiel:

```
mkdir -m 444 -p /home/myuser/Subverz1/Verz2
```

Legt das Verzeichnis »Verz2« nur mit Leserechten an; falls das Verzeichnis »Subverz1« nicht vorhanden ist, wird auch dieses angelegt.

Beachten Sie, daß es die Kurzform »md« von MS-DOS für diesen Befehl unter Unix nicht gibt.

11.2.8 Verzeichnis löschen

***rmdir** Verzeichnis...*

Löscht ein Verzeichnis. Beachten Sie, daß es den Kurzbefehl »rd« zu *rmdir* von DOS/Windows unter Unix nicht gibt.

11.2.9 Pfadnamen des Verzeichnisses anzeigen

pwd

Zeigt den Pfadnamen des aktuellen Verzeichnisses an. Dieser Befehl entspricht dem DOS-Befehl »cd« ohne Parameter. *cd* unter Unix wechselt aber im Gegensatz zum DOS-Befehl in das aktuelle Heimatverzeichnis des Anwenders.

Bei einigen Unix-Shells ist es nicht möglich, im Prompt den aktuellen Pfad anzeigen zu lassen. Da Unix-Pfade aber sehr lang sein können, ist *pwd* ein sehr nützlicher Befehl für die Orientierung im Verzeichnisbaum.

11.2.10 Dateien suchen

Für die Suche nach Dateien stehen – je nach genauer Aufgabe – einige Befehle zur Verfügung.

***find** Pfad... [Bedingung...]{Aktion}*

Sucht Dateien anhand eines regulären Ausdrucks und verarbeitet den Namen.

Bedingung:

-name Datei...	Dateiname als regulärer Ausdruck.
-perm Oktalzahl	Zugriffsrechte oktal für die gesuchten Dateien.
-type [b|c|d|f|l]	Dateityp Gerät, Byte (Programmdatei), Verzeichnis, normale Datei, Link-Datei.

-user Benutzername	Eigentümer.
-group Gruppenname	Gruppe.
-user Benutzername	Eigentümer.
-size Größe	Größe in Blöcken.
-atime Tage	Letzter Zugriff vor x Tagen.
-newer Datei	Dateien, die jünger als die angegebene sind, werden gesucht.

Aktion:

-exec Kommando	Unix-Kommando.
-ok Kommando	Unix-Kommando mit vorheriger Bestätigung.
-print	Gibt die Pfadnamen der gefundenen Dateien aus.

Beispiele:

```
find / -name datei1.* -print > aus1.txt
```

Gibt die Pfadnamen aller Dateien mit den Namen »datei1« und beliebiger Extension vom Unix-Wurzelverzeichnis aus und schreibt sie in die Ausgabedatei »aus1.txt«.

```
rm ` find / -name core -print `
```

Beseitigt alle »core«-Dateien, allerdings kann hier die Pipe keine großen Ergebnismengen bewältigen.

```
find / -name core -exec rm {} /;
```

Das gleiche Kommando, das hier aber immer zum Erfolg führt.

whereis [-bms] [-u] [-BMS -dir -... -f] Programm ...

Das Hilfsprogramm *whereis* ermittelt die Position ausführbarer Programme. Es zeigt nach dem Aufruf die Position des Programms und der zugehörigen Manpage an. Im Gegensatz zu *locate* wird keine Datenbank im Hintergrund ausgelesen.

Das Programm kennt die folgenden Parameter:

-B	Angabe der Verzeichnisse, die nach Binärdateien durchsucht werden sollen. Hierfür muß auch der Parameter »-f« angegeben sein.
-M	Angabe der Verzeichnisse, die nach Manpages durchsucht werden sollen. Hierfür muß auch der Parameter »-f« angegeben sein.
-S	Angabe der Verzeichnisse, die nach Programm-Quelltexten durchsucht werden sollen. Hierfür muß auch der Parameter »-f« angegeben sein.
-b	Suche nach Binärdateien.
-f	Beschränkt die Liste der Verzeichnisse nach »-B», »-M« oder »-S options«.
-m	Suche nach Manpages.
-s	Suche nach Quelltext-Verzeichnissen.
-u	Suche nach »ungewöhnlichen« Einträgen, das heißt nach Dateien, die keinem der definierten Typen entsprechen.

which [-as] [Befehl]

which übernimmt eine Liste von Befehlsnamen und durchsucht den Pfad nach den ausführbaren Dateien, die beim Aufruf dieses Befehls verwendet würden.

Die folgenden Optionen sind möglich:

-a	Liste aller Instanzen der ausführbaren Dateien anstelle von jeweils nur einer.
-s	Keine Bildschirmausgabe, stattdessen wird 0 gemeldet, wenn beliebige der angegebenen Dateien gefunden wurden, und 1, wenn die Suche fehlschlug.

Einige Shells besitzen einen eingebauten *which*-Befehl, der ähnlich oder gleich wie dieser Befehl arbeitet.

locate [-d database] Suchbegriff

locate findet schnell Dateien. Dazu wird eine Datenbank »/var/db/locate.database« durchsucht. Normalerweise sollte diese Datenbank, wenn *locate* öfters verwendet wird, wöchentlich aktualisiert werden.

Die Parameter sind:

-d database	Angabe einer alternativen Datenbank »database«.

Der Aufruf von *locate* macht nur Sinn, wenn die Datenbank auf einem aktuellen Stand ist. Das Programm zur Aktualisierung sollte also entweder beim Systemstart oder über »/etc/periodic/weekly/310.locate« gestartet werden oder bei Bedarf manuell. Dazu muß auf der Kommandozeile folgendes eingegeben werden:

```
/usr/libexec/locate.updatedb
```

11.2.11 Dateien konvertieren und kopieren, Disketten kopieren

***dd** [--help] [--version] [if=file] [of=file] [ibs=byte] [obs=byte] [bs=byte] [cbs=byte] [skip=blocks] [seek=blocks] [conv=typ]*

dd kopiert Daten normalerweise vom Standard-Input zum Standard-Output, hierbei können Blockgrößen und zusätzliche Verarbeitungsanweisungen angegeben werden.

Wenn der Parameter *conv* angegeben ist, muß dafür einer der folgenden Typen verwendet werden:

ascii	EBCDIC- nach ASCII-Umwandlung.
ebcdic	ASCII- nach EBCDIC-Umwandlung.
ibm	Konvertiert ASCII ins alternative EBCDIC-Format.
block	Konvertiert Records mit Zeilenende-Trennung in die im Parameter *cbs* angegebene Größe, wobei die Zeilenumbrüche in entsprechend viele Leerzeichen umgewandelt werden.
unblock	Umgekehrt wie *block*.
lcase	Wandelt Groß- in Kleinbuchstaben um.
ucase	Wandelt Klein- in Großbuchstaben um.
swab	Kippt zwei folgende Bytes um.
noerror	Nach Lesefehlern wird fortgesetzt.
notrunc	Die Ausgabedatei wird nicht abgeschnitten.
sync	Jeder Eingabeblock wird auf die Größe des Parameters *ibs* gesetzt, wobei mit 0x00-Zeichen aufgefüllt wird.

Die Namen der Parameter zeigen aber schon, was man im Grunde mit dem Befehl tun kann: die Konvertierung von Text in Groß- oder Kleinbuchstaben. *dd* ist aber auch in anderer Hinsicht ein wichtiges Programm. Man kann über die Angabe der Dateinamen (*if=Quelle*, *of=Ziel*) die Operationen auf normale Dateien und auch auf Devices (Geräte) ausführen. *dd* wird damit als ein Werkzeug zum Erzeugen und Zurückschreiben von Disketten-Images und damit wieder für das Kopieren von Disketten verwendet.

Um das Diskettenimage »boot1.flp« auf das Diskettenlaufwerk »/dev/fd0« (das dem DOS-Laufwerk A: entspricht) direkt auf das Device zu schreiben, muß *dd* folgendermaßen aufgerufen werden:

```
dd conv=notrunc bs=512 if=<Pfad>/boot1.fs of=/dev/fd0a
```

Der Austausch der Parameter *of* und *if* erzeugt übrigens das Diskettenimage. Beachten Sie, daß das Diskettenlaufwerk dafür nicht in das Dateisystem eingebunden sein braucht.

Weitere Kurzinformationen über *dd* erhalten Sie, wenn Sie das Programm (wie bei allen GNU-Utilities üblich) mit *--help* aufrufen. Wie bei den GNU-Utilities ebenfalls üblich, ist aber die Manpage äußerst knapp gehalten.

11.3 Wer darf was mit wem?

Bei einem Multi-User-Betriebssystem wie Unix ist es notwendig, Daten vor unliebsamen Nachbarn zu schützen. So darf es beispielsweise keinem anderen Benutzer als dem Systemadministrator erlaubt sein, auf Systemdateien zuzugreifen. Benutzer sind Gruppen zugeordnet. Beim Zugriff auf eine Datei wird unterschieden, ob der zugreifende der Besitzer der Datei, in einer Gruppe des Besitzers, oder aber ein völlig Fremder ist. Dabei wird beim Festlegen der Rechte in folgender Reihenfolge vorgegangen: Ein Benutzer kann also sich selbst den Zugriff auf eine Datei sperren, aber dem Rest seiner Arbeitsgruppen freigeben. Eine Sonderrolle nimmt auch das sogenannte »User-ID-Bit« ein. Über dieses Bit bleiben die Zugriffsrechte des Besitzers einer Programmdatei an den Benutzern eines Programmes »haften«. Dies ist beispielsweise dann sinnvoll, wenn Benutzern der Zugriff auf eine Datenbank nur über das entsprechende Programm ermöglicht werden soll.

11.3.1 Benutzergruppen-Attribut einer Datei ändern

chgrp *Gruppe* {*Datei...* | Verzeichnis}

Ändert das Attribut »Benutzergruppe« einer Datei und/oder eines Verzeichnisses.

Beispiel:

```
chgrp abteilung5 datei1 verzeichnis1
```

Ändert den Gruppennamen von »datei1« und »verzeichnis1« in »abteilung5«.

11.3.2 Eigentümer-Attribut ändern

***chown** [-R] Eigentümer[:Gruppe] {Datei... | Verzeichnis...}*

Ändert das Attribut des Eigentümers und – falls mit angegeben – auch der Gruppe einer Datei und/oder eines Verzeichnisses.

Parameter:

-R	Rekursiv, ändert die Eigentümerattribute auch in allen Unterverzeichnissen und den darin enthaltenen Dateien.
:Gruppe	Ändert gleichzeitig auch das Gruppenattribut. Zwischen Eigentümername und Gruppenname darf kein Leerzeichen stehen.

Beispiel:

```
chown -R user1:abteilung1 verzeichnis1
```

Ändert von »verzeichnis1« und allen enthaltenen Unterverzeichnissen inklusive aller Dateien den Eigentümer in »user1« und die Gruppe in »abteilung1« um.

11.3.3 Zugriffsrechte ändern

***chmod** {Oktalzahl} | {[-R] {[ugo] | [a]}{+|-}{rwxt}} Datei... | Verzeichnis...}*

Ändert die Zugriffsrechte von Dateien und Verzeichnissen. Die Parameter:

Über die Kombination der Oktalzahlen:

4000	Setzt bei der Ausführung die Eigentümer-ID des Dateieigentümers als effektive ID ein (s-Bit).
2000	Setzt bei der Ausführung die Gruppen-ID des Dateieigentümers als effektive ID ein (s-Bit).
1000	Sticky-Bit (auch »Save Text Bit« genannt) ist Programmcode, der von mehreren Benutzern geteilt werden kann, verbleibt auch nach Beendigung des Programms im Hauptspeicher oder Swap-Bereich.
0400	Setzt das Leserecht für den Eigentümer.
0200	Setzt das Schreibrecht für den Eigentümer.
0100	Setzt das Ausführungsrecht für den Eigentümer.

0040	Setzt das Leserecht für die Gruppe.
0020	Setzt das Schreibrecht für die Gruppe.
0010	Setzt das Ausführungsrecht für die Gruppe.
0004	Setzt das Leserecht für andere Benutzer.
0002	Setzt das Schreibrecht für andere Benutzer.
0001	Setzt das Ausführungsrecht für andere Benutzer.

Die Oktalzahlen sind beliebig aufaddierbar, um Kombinationen der unterschiedlichen Rechte in einem Kommando setzen zu können.

u	Eigentümer.
g	Gruppe.
o	Andere Benutzer.
a	Eigentümer, Gruppe und andere Benutzer auf einmal.
+	Vergeben von Rechten.
-	Entziehen von Rechten.
r	Leserecht.
w	Schreibrecht.
x	Ausführungsrecht von Dateien, auch Durchsuchen von Verzeichnissen und Anlegen von Dateien in Verzeichnissen.
s	Setzt Eigentümer- oder Gruppen-ID bei der Ausführung.
g	Gruppe.
o	Andere Benutzer.
t	Sticky-Bit.

Beispiele:

```
chmod ug-w+r datei1
```

Dem Eigentümer und der Gruppe von »datei1« wird das Schreibrecht entzogen und das Leserecht erteilt.

```
chmod 755 verzeichnis1
```

Von »verzeichnis1« wird dem Eigentümer das Lese-, Schreib- und Ausführungsrecht und der Gruppe beziehungsweise allen anderen das Lese- und Ausführungsrecht erteilt.

umask *{Oktalzahl}*

Setzt die Voreinstellungen für die Zugriffsrechte neu angelegter Dateien. Dabei entspricht die anzugebende Oktalzahl dem invertierten Wert der gewünschten Zugriffsrechte. Es werden also die Bits angegeben, die nicht gesetzt werden sollen.

Beispiel:

```
umask 022
```

Setzt die Zugriffsrechte neu angelegter Dateien auf 755.

11.4 Bearbeitung von Textdateien

Bei der Ausgabe eines Textfiles sollte sichergestellt werden, daß es sich auch wirklich um eine Textdatei handelt. Binärdateien können unter Umständen Steuerzeichen enthalten, die Terminalparameter beeinflussen. Um den Typ einer Datei zu ermitteln, bedient man sich des Programms *file*.

11.4.1 Dateityp ermitteln

file *[Dateiname]*

file unterscheidet zwischen den Typen:

ascii text	Textdatei, die auch Sonderzeichen enthalten kann.
postscript	Datei mit Postscript-Anweisungen.
block special	Blockorientiertes Gerät, beispielsweise CD-Laufwerk oder Festplatte.
character spezial	Zeichenorientiertes Gerät, beispielsweise serielle Schnittstelle.
commands text Shell	Kommandoprozedur.
data	Binärdatei.
directory	Verzeichniseintrag.
empty	Leere Datei.
executable	Ausführbare Datei mit Programmcode.

11.4.2 Textdatei auf Bildschirm ausgeben

Wie gebe ich eine Textdatei auf dem Bildschirm aus? Nun, am einfachsten geschieht dies mit dem Kommando *cat*.

***cat** [Dateiname]*

Zeigt die Inhalte von Dateien auf dem Bildschirm an. Ist die Datei allerdings größer als das aktuelle Terminal, verschwindet der Text am oberen Rand. Um dies zu verhindern, kann die Ausgabe mit ^*s* angehalten und mit ^*q* fortgesetzt werden.

11.4.3 Inhalt seitenweise anzeigen

Um größere Texte einzusehen, ist es bequemer, das Kommando *more* zu verwenden.

***more** [Dateiname]*

Zeigt den Inhalt einer Textdatei seitenweise an. Seitenweises Blättern erfolgt mit der Leertaste. Um sich zeilenweise im Text abwärts zu bewegen, betätigt man die Eingabetaste. Die Ausgabe kann mit Druck auf die Taste [q] abgebrochen werden.

11.4.4 Teile einer Datei anzeigen

Spezieller sind die Befehle *head* und *tail*. Mit diesen Kommandos ist es möglich, nur den ersten beziehungsweise den letzten Teil einer Datei auszugeben.

***head** -[Zeilennummer] [Datei]*

Gibt den Inhalt einer Textdatei bis zu einer bestimmten Zeilennummer aus.

Beispiel:

```
head -3 ~/.profile
```

Gibt die ersten drei Zeilen Ihrer persönlichen *bash*-Konfigurationsdatei aus.

```
ls -al | head -5
```

gibt die ersten fünf Einträge im aktuellen Verzeichnis aus.

***tail** +[Zeilennummer] [Datei]*

Gibt den Inhalt einer Textdatei ab einer bestimmten Zeilennummer aus.

***tail** -[Anzahl] [Datei]*

Gibt die letzten Zeilen einer Textdatei aus.

Beispiel:

```
ls -al | tail -1
```

Gibt den letzten Eintrag im aktuellen Verzeichnis aus.

```
ls -al | head -10 | tail +3
```

Gibt die Einträge 3 bis 10 im aktuellen Verzeichnis aus.

11.4.5 Ausgabe in Datei und Standardausgabe zugleich umleiten

tee [-a -i] [Datei]

Leitet die Standardeingabe auf eine Datei und gleichzeitig wieder auf die Standardausgabe um.

Die Parameter:

-a	Hängt an die Datei an, statt sie zu überschreiben.
-i	Ignoriere das Signal *INT*, das den Job sonst in den Hintergrund senden würde.

Beispiel:

```
cat /etc/rc.conf | tee myrc
```

Zeigt die Datei »/etc/rc.conf« an und kopiert sie gleichzeitig in die Datei »myrc«.

11.4.6 Durchsuchen einer Datei nach Zeichenketten

strings [-a -f -v -n Länge -t {o,x,d}] [Datei]

Durchsucht eine angegebene Binärdatei nach zusammenhängenden Zeichenketten.

Die Parameter:

-a	Durchsucht nicht nur bestimmte Sektionen einer ausführbaren Datei, sondern alles.
-f	Druckt vor jede Zeichenkette den Dateinamen.
-v	Gibt die Versionsnummer aus.

-n	Setzt die Mindestlänge der zu findenden Zeichenkette auf *n* Buchstaben. Voreinstellung ist 4.
-t	Gibt die Position der gefundenen Zeichenfolge innerhalb der Datei in Oktal (o), Hexadezimal (x) oder Dezimal (d) an.

Beispiel:

```
strings -a -n 15 /usr/local/bin/bash
```

Gibt alle Zeichenketten der Datei »/usr/local/bin/bash«, die mehr als 15 Buchstaben umfassen, aus.

11.4.7 Textdateien vergleichen

***diff** [-w -i -e] Datei1 Datei2*

Vergleicht zwei Textdateien und gibt deren unterschiedlichen Zeileninhalte aus.

Die Parameter:

-w	Ignoriert Tabulatoren und Leerzeichen im Text.
-i	Ignoriert die Groß-/Kleinschreibung im Text.
-e	Erstellt eine Ausgabe, mit der der Editor *ed* »Datei2« aus der »Datei1« wieder erzeugt.

Beispiel:

```
diff -w datei1 datei2 > ausgabe.dif
```

Gibt die unterschiedlichen Zeilen von Datei »datei1« und »datei2« in die Ausgabedatei »ausgabe.dif« aus.

11.4.8 Textdatei erstellen

Wie erstellen Sie nun eine Textdatei? Zum einen können Sie natürlich einen Editor verwenden. NetBSD bietet verschiedene Editoren, wie beispielsweise »vi«, »ed«, »emacs« oder »joe« an. Es geht aber auch einfacher. Wollen Sie nur eine kurze Textdatei erzeugen oder einige Zeilen an eine bestehende Datei anhängen, können Sie die Ausgabe von *cat* in eine Datei umleiten. Beachten Sie aber, daß das Dateiende in Unix durch [Strg]-[D] und nicht, wie bei MS-DOS, mit [Strg]-[Z] signalisiert wird.

Beispiele:

```
cat > dateineu
```

Leitet die Ausgabe des Terminals in eine neue Datei um.

```
cat >> dateineu
```

Leitet die Ausgabe des Terminals in eine neue Zeile einer bestehenden Datei um.

Beispiel:

```
cat > .forward
myadress@somewhere.intheworld
^d
cat >> .forward
myotheradress@somewhere.else
^d
```

Diese Eingabe erzeugt eine Datei namens ».forward« im aktuellen Verzeichnis. Durch den führenden Punkt ist die Datei versteckt und wird bei einem einfachen *ls* nicht angezeigt. Was enthält diese Datei nun?

```
cat .forward
```

erzeugt als Ausgabe:

```
myadress@somewhere.intheworld
myotheradress@somewhere.else
```

Dieses Beispiel ist nicht aus der Luft gegriffen. Falls die Datei nun in Ihrem Home-Directory steht, wird jede ankommende E-Mail automatisch an die entsprechenden Adressen weitergeleitet. Dieser Vorgang wird »forwarding« genannt.

Das Anlegen einer Textdatei kann alternativ aber auch über die Umleitung des Befehls *echo* vorgenommen werden, der später noch beschrieben wird. Eine leere Datei legt man am bequemsten mit *touch* an.

11.4.9 Dateien durchsuchen

***grep** [-c -i -v -w] Muster... [Datei...]*

Durchsucht Dateien nach einem Suchmuster und gibt die entsprechenden Zeilen aus.

Parameter:

-c	Gibt als Ergebnis nur die Anzahl der Zeilen aus, in denen das Suchmuster gefunden wurde.
-i	Groß- und Kleinschreibung werden bei der Suche nicht berücksichtigt.
-v	Gibt alle Zeilen aus, in denen das Suchmuster nicht gefunden wurde.
-w	Führt eine Wortsuche durch.
Muster	Es können begrenzte reguläre Ausdrücke verwendet werden.

Beispiele:

```
grep "^[a-zA-Z0-9]" datei1
```

Gibt von »datei1« alle Zeilen aus, bei denen an der ersten Stelle ein Buchstabe oder eine Zahl steht.

```
grep -v "^#" datei1
```

Gibt von »datei1« alle Zeilen aus, die an erster Stelle kein »#«-Zeichen stehen haben.

11.5 Manipulation von Dateien

Nun können Sie sich also schon sehr gut im Verzeichnisbaum bewegen und mit Textdateien umgehen. Sie werden feststellen, daß Unix zur Dateimanipulation noch eine ganze Reihe weiterer Möglichkeiten bietet. Besonders interessant sind dabei die sogenannten »Links«. Es handelt sich hierbei um logische Verknüpfungen, gewissermaßen um einen Zeiger zu einer Datei. Dieses Prinzip wird unter Unix sehr oft verwendet, um Verzeichnisse, beispielsweise aus Kompatibilitätsgründen, unter verschiedenen Namen einzublenden.

11.5.1 Dateien und Verzeichnisse kopieren

***cp** [-R -H-L - P -f -i -p] Quelle Ziel*

Kopiert Dateien oder Verzeichnisse. Ist die Quelldatei im Zielverzeichnis vorhanden, wird sie überschrieben.

Die wichtigsten Parameter:

-R	Rekursives Kopieren von Dateien und Verzeichnissen in das Zielverzeichnis.
-i	Ist die Zieldatei angelegt, wird gefragt, ob sie überschrieben werden soll.
-p	Zeitstempel und Zugriffsrechte bleiben erhalten.

-f	Man kann festlegen, daß beim Kopieren von Dateien grundsätzlich vor dem Überschreiben nachgefragt wird, ob dies gewünscht ist (Parameter *-i*). Mit dem Parameter *-f* werden diese Nachfragen unterbunden, also der Parameter *-i* überschrieben.

Beispiele:

```
cp altedatei.txt neuedatei.txt
```

Legt eine Dateikopie von »altedatei.txt« im aktuellen Verzeichnis mit dem Namen »neuedatei.txt« an.

```
cp datei.txt /sub/sub/verzeichnis1
```

Kopiert »datei.txt« in das Verzeichnis »/sub/sub/verzeichnis1«.

```
cp -R /sub/sub/verzeichnis1 /unter/unter/verzeichnis2
```

Kopiert alle Dateien und Verzeichnisse aus »/sub/sub/verzeichnis1« in das Verzeichnis »/unter/unter/verzeichnis2«.

11.5.2 Dateien und Verzeichnisse löschen

rm [-f | -i] [-d -r -P -R -r -W] {Datei... | Verzeichnis...}

Löscht Dateien oder Verzeichnisse oder hebt Links zu ihnen auf.

Die wichtigsten Parameter:

-f	*force*; sollen schreibgeschütze Dateien gelöscht werden, wird keine explizite Bestätigung gefordert.
-i	*interactive*; jede einzelne Datei muß vor dem Löschen vom Benutzer bestätigt werden.
-r	*recursive*; löscht alle Verzeichnisse und deren Inhalte rekursiv.
-d	*directory*; löscht ein Verzeichnis.

Beispiele:

```
rm datei1
```

löscht die Datei »datei1«.

```
rm -r -i verz1
```

löscht das Verzeichnis »verz1« und verlangt für jede Datei und jedes Unterverzeichnis eine Bestätigung vom Benutzer.

11.5.3 Links erzeugen

***ln** [-s] {Quellendatei[Zieldatei] | Quellendatei... Zielverzeichnis}*

Erzeugt eine logische Kopie (Link) der Quelldatei, wobei beide im selben Dateisystem liegen müssen. Ausnahme: *-d* ermöglicht einen Debugmodus.

Parameter:

-s	Erstellt eine symbolische Verbindung zwischen Quell- und Zieldatei.

Physikalisch werden nicht diese Dateien kopiert, sondern nur die Verweise in den entsprechenden Verzeichniseinträgen. Die physische Datei wird erst dann gelöscht, wenn der letzte logische Verweis (Link) gelöscht wurde.

Beispiele:

```
ln datei1 datei2
```

Erzeugt eine logische Kopie von Datei »datei1«. Existiert »datei2« noch nicht, wird sie erzeugt. Ist »datei2« bereits vorhanden, wird ihr Verweis auf »datei1« umgesetzt.

```
ln datei1 verzeichnis2
```

Erzeugt eine logische Kopie von Datei »datei1« im Verzeichnis »verzeichnis2« mit demselben Namen von »datei1«.

11.5.4 Dateien und Verzeichnisse verschieben

***mv** [-i -f] {{Datei... | Verzeichnis...} Verzeichnis | {Datei | Verzeichnis} NeuerName}*

Verschiebt Dateien oder Verzeichnisse beziehungsweise benennt sie um.

Parameter:

-i	Falls die Zieldatei oder das Zielverzeichnis schon vorhanden ist, wird bei dieser Option vor dem Überschreiben vom Anwender eine Bestätigung mit [j] gefordert.
-f	Es wird vor dem Überschreiben keine Bestätigung vom Anwender gefordert.

Beispiele:

```
mv datei1 datei2
```

»datei1« wird in »datei2« umbenannt.

```
mv verzeichnis1 verzeichnis2
```

Ist »verzeichnis2« nicht vorhanden, wird »verzeichnis1« in »verzeichnis2« umbenannt. Ansonsten wird »verzeichnis1« ein Unterverzeichnis von »verzeichnis2«.

```
mv datei1 verzeichnis2
```

»datei1« wird in das Verzeichnis »verzeichnis2« verschoben.

```
mv datei1 verzeichnis2/datei2
```

»datei1« wird in das Verzeichnis »verzeichnis2« verschoben und in »datei2« umbenannt.

11.5.5 Dateidatum modifizieren

***touch** [-a -c -f -h -m] [-r Datei] [Zeit] {Datei... | Verzeichnis...}*

Verändert das Modifikationsdatum von Dateien oder Verzeichnisse auf das aktuelle oder angegebene Datum. Gibt es die Datei nicht, wird sie angelegt.

Die wichtigstenParameter:

-a	Ändert nur das Datum des letzten Zugriffs.
-c	Existiert die angegebene Datei nicht, wird sie auch nicht angelegt.
-f	Versucht, das Datum auch bei nicht existierenden Schreib/Leserechten zu ändern.
-m	Ändert nur die Zeit des letzten Zugriffs.
Zeit	Die Zeitangabe hat das Format »MMTTHHmm« (Monat, Tag, Stunde, Minute).

Beispiel:

```
touch -c 07042135 datei1
```

Wenn »datei1« existiert, wird das Modifikationsdatum auf den 7. April 21:35 geändert.

11.6 Leben im Netz

Unix ist speziell für den Netzwerkeinsatz geschaffen. Daher besitzt es auch eine ganze Reihe von Befehlen, die zur Kommunikation und Datenübertragung innerhalb des Netzes dienen.

11.6.1 Dateien übertragen

***ftp** [-d -g -i -v] [Hostname]*

Überträgt Dateien zwischen dem lokalen und dem entfernten Host.

Parameter:

-d	Ermöglicht einen Debugmodus.
-g	Dateinamen werden nicht expandiert.
-i	Verhindert die Bestätigungsaufforderung beim Übertragen mehrerer Dateien.
-v	Zeigt die Antworten des entfernten Hosts an und gibt die Transferstatistiken aus.

Beispiel:

```
ftp -giv myhost
```

Es wird eine Dateitransferverbindung zum Rechner »myhost« aufgebaut.

11.6.2 Nachricht auf Terminal übermitteln

***write** [Benutzer]*

Übermittelt dem angerufenen Benutzer eine Nachricht auf das aktuell verwendete Terminal.

Beispiel:

```
write root
Hallo Chef, immer noch am arbeiten?
^d
```

Gibt dem Administrator die entsprechende Nachricht am Bildschirm aus. Die Nachricht wird Zeile für Zeile übertragen, bis das Programm mit [Strg]-[D] beendet wird. Der angesprochene Benutzer kann währenddessen weiterarbeiten oder aber auch seinerseits mit *write* antworten.

11.6.3 Bildschirmmeldungen ausschalten

mesg [y]|[n]

Schaltet Bildschirmmeldungen anderer Benutzer an oder aus.

Beispiel:

```
# mesg n
```

Write-Nachrichten, abgesehen von denen des Administrators, sind nun gesperrt.

```
# mesg
is n
# mesg y
```

Write-Nachrichten sind nun wieder möglich.

11.6.4 Chat-Verbindung herstellen

talk [Benutzer@Adresse]

Öffnen einer Chat-Verbindung zu einem anderen Benutzer im Netzwerk. Der angesprochene Benutzer wird auf seinem Terminal aufgefordert, auf den Request mit einem entsprechenden *talk*-Befehl zu antworten.

Beispiel:

```
talk meyer@ rhds01.rz.fht-esslingen.de
```

Öffnet eine talk-Verbindung zu Benutzer »meyer« auf dem Rechner mit der Adresse »rshx01.rz.fht-esslingen.de«.

11.6.5 Systemweite Meldungen versenden

wall

Spezielles Write an alle Benutzer des Systems. Besonders nützlich für Warnungen des Administrators.

11.6.6 Job Control

Das System vergibt jedem vom Benutzer gestarteten Prozeß (Job) eine eigene Nummer. Mit den Befehlen *wait, kill, fg* oder *bg* und der Jobnummer können die Jobs manipuliert werden.

Mit [Strg]-[Z] werden im Vordergrund laufende Prozesse unterbrochen. Für den unterbrochenen Job wird die Jobnummer ausgegeben. Ein Kommando kann durch Anhängen des Zeichens »&« gleich in den Hintergrund geschickt werden. Für die Bezugnahme auf einen Job stehen folgende Möglichkeiten zur Verfügung:

Jobausführung fortsetzen

***bg** [job...]*

Setzt die Ausführung unterbrochener Jobs im Hintergrund fort.

Job in Vordergrund holen

***fg** [job...]*

Holt einen Job wieder in den Vordergrund zurück.

Aktuelle Jobs ausgeben

***jobs** [-l -p][job...]*

Gibt die aktuellen Jobs aus. Parameter:

-l	Gibt zusätzliche Informationen zu den Jobs aus.
-p	Ausgabe der Prozeß-ID der Jobs.

Job-Beendigung abwarten

***wait** [job...]*

Die Beendigung des Jobs wird abgewartet.

11.7 Sonstige Kommandos

11.7.1 Infos über System anzeigen

***uname** [-a]|[-m -n -r -s -v]*

Zeigt aktuelle Informationen zum Betriebssystem und dem Prozessor an.

Parameter:

-a	Entspricht dem Aufruf von *uname -mnrsv*.
-m	Rückgabe der Maschinen-ID, bei NetBSD des Namens des Ports.
-n	Knotenname (Hostname).
-r	Release-Nummer des Betriebssystems, bei NetBSD 1.6 der Wert »1.6«
-s	Der Systemname, bei NetBSD die Zeichenkette »NetBSD«.
-v	Der Name und die Version des Kernels und aus welcher Datei der Kernel erzeugt wurde.

Beispiel:

```
uname -s
NetBSD
```

11.7.2 Angemeldete Benutzer zeigen

who [Datei] | who am i

Zeigt die zur Zeit im System angemeldeten Benutzer an.

Beispiel:

```
who am i
```

Zeigt den eigenen Benutzernamen an.

Es gibt tatsächlich auch noch ein weiteres Programm *whoami* (dieses Mal zusammengeschrieben), das ebenfalls sagt, wer man ist. Die beiden Aufrufe liefern aber unterschiedliche Resultate:

```
root@beastie: ~# /usr/sbin#whoami
root
root@beastie: ~# /usr/sbin#who am i
joerg           ttyp1   Jul 20 19:50   (192.168.10.12:0.)
root@beastie: ~# /usr/sbin#
```

Beachten Sie den Unterschied in der Aussage zum ge-*su*-ten Anwender.

Für die Ermittlung der Anwender stehen in Form von *lastlogin* und *last* außerdem weitere interessante Befehle zur Verfügung. *lastlogin* gibt an, wann sich die aktuell im System befindlichen User eingeloggt haben, *last* die letzten Logins der User und deren ttys.

11.7.3 Datum anzeigen

date

Zeigt das aktuelle Datum und die aktuelle Zeit an. Außerdem wird mit diesem Befehl auch die aktuelle Uhrzeit gesetzt.

```
date 200301122001
```

setzt die Uhrzeit auf den 12. Januar 2003 um 20:01 Uhr.

11.7.4 Zeichenfolge ausgeben

echo [-n] "Text"

Gibt eine Zeichenfolge aus.

-n	Nach Ausgabe der Zeichenfolge erfolgt kein Zeilenumbruch.
Text	Zeichenfolge.

Escape-Sequenzen, die in der Zeichenfolge verwendet werden können:

\b	Backspace, ein Zeichen zurück
\c	Keine neue Zeile
\f	Form-Feed
\n	Neue Zeile
\r	Carriage Return
\t	Tabulator
\v	Vertikaler Tabulator
\\	Backslash
\0*n*	Oktalzahl

Beispiel:

```
echo "Hallo Welt"
```

11.8 Datensicherung

11.8.1 Dateien komprimieren

Unix kennt mehrere Kompressionsprogramme. Standardmäßig eingesetzt wird das Paket *compress/uncompress*. Wichtig sind außerdem die beiden Pakete *gzip/gunzip* und *bzip2/bunzip2*. Selbstverständlich gibt es für NetBSD auch andere Programme für andere Archivformate wie ».zip«, ».lzh« und ».arj«. Diese Programme sind aber niemals als Standard beigefügt und spielen auch nicht die Rolle wie unter OS/2 und Windows.

compress [-v -c] Datei...

Komprimiert Dateien, wobei die angegebene Datei in eine neue Datei mit dem Suffix ».Z« umbenannt wird. Die Parameter:

-v	»Geschwätziger« Modus, gibt die Komprimierung in Prozent aus.
-c	Die komprimierte Datei wird in die Standardausgabe geschrieben, es werden keine Dateien geändert.

Beispiel:

```
compress datei1.suf datei2.suf
```

Komprimiert die Dateien »datei1.suf« und »datei2.suf« und legt das Ergebnis in der Datei »datei1.suf.Z« ab.

gzip [cdfhlLrt19] [-s suffix] [Datei...]

Packt Dateien nach einem moderneren Verfahren als *compress*. Im Gegensatz zu den bekannten MS-DOS-Komprimierungsprogrammen faßt *gzip* die Dateien nicht zusammen, sondern hängt nur die Extension ».gz« an die einzelnen Dateien an. Daher wird dieses Tool gerne in Verbindung mit *tar* verwendet. Zuerst werden die Dateien mit *tar* archiviert, und dann das Archiv mit *gzip* gepackt. Anstatt der so entstehenden Extension ».tar.gz« wird oft einfach ».tgz« verwendet. Solche Dateien werden von *gunzip* und *gzip -d* automatisch wieder in ».tar«-Dateien verwandelt.

Parameter:

-c	Gibt das Ergebnis auf der Standard-Ausgabe aus und ändert nicht die Originaldateien.
-d	Entpacken einer Datei.
-f	Force-Modus. Dateien werden überschrieben und Links komprimiert.

-h	Hilfe.
-l	Inhalt des Archivs anzeigen.
-L	Lizenzinformationen ausgeben.
-q	Quiet-Modus. Warnungen werden unterdrückt.
-r	Unterverzeichnisse mitpacken.
-s .suf	Suffix ».suf« anhängen.
-t	Archiv auf Fehler testen.
-1	Schnelle Komprimierung.
-9	Maximale Komprimierung.

Beispiele:

```
gzip mytext.txt
```

Verwandelt die Datei »mytext.txt« in eine gepackte Datei namens »mytext.txt.gz«.

```
gzip -d mytext.txt.gz
```

Verwandelt die Datei »mytext.txt.gz« wieder zurück in eine Datei namens »mytext.txt«.

bzip2 [-cdfkqstvzVL123456789] [Datei(en) ...]

bzip2 packt Dateien mit der hocheffizienten Burrows-Wheeler-Blocksortierung und Huffman-Kodierung. Die Parameter sind sehr ähnlich zu GNU *gzip*, aber nicht identisch. *bzip2* erwartet eine Liste von Dateinamen. Diese werden komprimiert und jeweils nach »Originaldatei.bz2« kopiert. Dabei wird in der Voreinstellung die Originaldatei gelöscht.

Die Parameter:

-c	Komprimieren oder Auspacken von oder in die Standard-Ein-/Ausgabe.
-d	Dekompression durchführen. *bzip2 -d* entspricht dem Aufruf von *bunzip2*.
-z	Kompression durchführen. *bunzip2 -z* entspricht dem Aufruf von *bzip2*.
-t	Integrität eines Archivs prüfen.
-f	Überschreiben von Ausgabedateien erzwingen. Normalerweise wird abgebrochen, wenn die Ausgabedatei bereits vorhanden ist.
-k	Eingabedatei nicht löschen.

-s	Speicherschonend arbeiten.
-q	Warnungen unterdrücken. Fehlermeldungen werden trotzdem ausgegeben.
-v	Zusätzliche Hinweise ausgeben.
-L	Ausgeben der Lizenzbedingungen von bzip2.
-V	Ausgeben der Versionsnummer des Programms.
-1 .. -9	Setzen der Blockgröße beim Komprimieren, diese Angabe ist beim Entpacken wirkungslos.

Das Programm *bunzip2* zum Entpacken der Dateien ist nur ein Link oder eine Kopie auf *bzip2*. Je nachdem, wie der Name beim Aufruf lautet, wird entschieden, ob ge- oder entpackt wird.

11.8.2 Dateien entpacken

uncompress [-f][-c]

Stellt die Originaldatei wieder her, die durch den Befehl *compress* komprimiert wurde.

Die Parameter:

-f	Erzwingt die Komprimierung oder Dekomprimierung.
-c	Die dekomprimierte Datei wird nach »stdout« geschrieben.

Beispiel:

```
uncompress datei2.suf.Z
```

Dekomprimiert die Datei »datei2.suf.Z« in die Datei »datei2.suf«.

gunzip [cfhlLrt] [-s suffix] [Datei...]

Verwandelt *gzip*-gepackte Dateien wieder zurück.

-c	Gibt das Ergebnis auf der Standard-Ausgabe aus und ändert nicht die Orginaldateien.
-f	Force-Modus. Dateien werden überschrieben.
-h	Hilfe.
-l	Inhalt des Archivs anzeigen.

-L	Lizenzinformationen ausgeben.
-q	Quiet-Modus. Warnungen werden unterdrückt.
-r	Unterverzeichnisse mitentpacken.
-s .suf	Suffix ».suf« anhängen.
-t	Archiv auf Fehler testen.

Beispiele:

```
gunzip mytext.txt.gz
```

Verwandelt die Datei »mytext.txt.gz« wieder zurück in eine Datei namens »mytext.txt«.

```
gunzip mytext.tgz
```

Verwandelt die Datei »mytext.tgz« zurück in ein tar-Archiv namens »mytext.tar«.

***bunzip2** [-fkvsVL] [Datei(en) ...]*

Entpacken von Dateien, die mit *bzip2* gepackt wurden. Die Parameter sind identisch.

11.8.3 Dateisicherung

***cpio** -i [cdfkmrsStuv][-E Datei] [-H hdr][-I Datei [-M Nachricht]] [Zeichenmuster]*

Parameter:

-i	Eingabemodus.
-o [aAcLv] [-H hdr] [-O Datei -M Nachricht]	Ausgabemodus.
-p [adlLmuv] Verzeichnis	Durchgangsmodus; schreibt Verzeichnis und Dateien in Archive und liest sie auch wieder aus. Die erste Option *-i, -o, -p* bestimmt den Modus, in dem das Kommando *cpio* verwendet wird.
-a	Setzt den Zeitstempel des letzten Zugriffes der entsprechenden Datei oder des Verzeichnisse zurück.
-A	Hängt die zu sichernden Informationen an ein bestehendes Archiv an. Diese Option kann nicht in Verbindung mit Bandarchiven verwendet werden.

-c	Die Headerinformationen des Archivs werden im ASCII-Format geschrieben.
-d	Beim Zurückschreiben von Dateien aus einem Archiv werden auch die entsprechenden Verzeichnisstrukturen mit angelegt.
-E *Datei*	Datei enthält eine Liste von Dateinamen, die zurückgesichert werden sollen.
-f	Schließt beim Sichern alle Dateien aus, auf die das Zeichenmuster zutrifft.
-H *hdr*	Zum Einlesen oder Schreiben von Fremdformaten wie zum Beispiel tar.
-I *Datei*	Die Headerinformationen des Archivs werden im ASCII-Format geschrieben.
-d	Beim Zurückschreiben von Dateien aus einem Archiv werden auch die entsprechenden Verzeichnisstrukturen mit angelegt.
-E *Datei*	Spezifiziert eine Datei als Eingabearchiv.
-k	Liest Datenträger auch mit nicht verwendbaren Headerinformationen, solange bis ein *cpio*-Header gefunden wird.
-l	In Verbindung mit dem Durchgangsmodus werden – wenn möglich – Dateien gelinkt anstatt kopiert.
-L	Beim Zurückschreiben von Dateien werden symbolische Links aufgelöst, es werden also nicht nur die gelinkten Dateien zurückgeschrieben, sondern auch die Originaldateien.
-m	Das Datum der letzten Änderung bleibt erhalten.
-M *Nachricht*	In Verbindung mit den Optionen -I und -O kann eine Nachricht mitgegeben werden, die ausgegeben wird, wenn das Medium gewechselt werden muß.
-O *Datei*	Schreibt die Ausgabe von *cpio* anstatt nach *stdout* direkt in eine Datei.
-r	Ermöglicht beim Zurücksichern von Dateien deren Umbennung.
-t	Gibt nur die Datei- und Verzeichnisnamen des Archivs aus.
-u	Läßt das Überschreiben neuerer Dateien über bestehende ältere Dateien zu.
-v	»Geschwätziger Modus«, kommentiert die einzelnen Aktionen beim Sichern oder Zurückschreiben.

Beispiele:

```
find /usr/local/testdaten -print | cpio -ov > /dev/rmt0
```

Sichert alle Dateien im Verzeichnis »/usr/local/testdaten« und den darunterliegenden Strukturen auf das Band im Gerät »rmt0«.

```
cpio -it -I archiv.dat
```

Zeigt alle Dateien und Verzeichnisse im Archiv »archiv.dat« an.

```
find /usr/local/testdaten -print | cpio -pdv /home/testuser
```

Kopiert alle Dateien und Verzeichnisse im Verzeichnis »/usr/local/testdaten« in das Verzeichnis »/home/testuser«.

11.8.4 Dateien sichern und zurückschreiben

***tar** {-c|-r|-t|-u|-x}[-v -w]-f Archiv [Datei... |Verzeichnis]*

Sichert Datei und Verzeichnisse auf ein Medium oder in eine Datei und schreibt sie auch wieder zurück.

Parameter:

-c	Erstellt ein neues Archiv.
-r	Fügt Dateien zu einem bestehenden Archiv hinzu.
-t	Zeigt alle Dateien und Verzeichnisse in einem Archiv an.
-u	Fügt Dateien einem Archiv hinzu, wenn sie dort noch nicht enthalten oder mittlerweile verändert wurden.
-x	Schreibt einzeln benannte Dateien zurück.
-v	»Geschwätziger Modus«, kommentiert die einzelnen Aktionen.
-w	Interaktiver Modus, wartet auf die Bestätigung des Benutzers.
-f Archiv	Bestimmt das zu verwendende Archiv, entweder ein Gerät oder eine Datei. Die Endung ».tar« wird nicht automatisch angehängt. Leerzeichen zwischen -f und dem Archivnamen werden ignoriert.
-z	Packt beziehungsweise entpackt die Datei via *gzip/gunzip* (Dateiendungen ».tar.gz« und .»tgz«). Besitzt die Datei die Endung ».bz2« oder ».tar.bz2«, wird bei NetBSD automatisch anstelle von *gunzip bunzip2* zum Entpacken verwendet.
-Z	Packt beziehungsweise entpackt die Datei via *compress/uncompress* (Dateiendung ».tar.Z«).

Beispiele:

```
tar -c -f /dev/rmt0 .
```

Schreibt alle Dateien und Unterverzeichnisse im aktuellen Verzeichnis auf das Band im Gerät »rmt0«.

```
tar -tv -f verzeichnis_b.tar
```

Zeigt alle Dateien und Verzeichnisse im Archiv »verzeichnis_b.tar« an.

```
tar -x -f mytext.tar
```

Entpackt alle Dateien aus der Datei »mytext.tar«.

Es gibt mehrere Versionen des Programms »tar«. Die obigen Parameter gelten für das bei NetBSD standardmäßig mitgelieferte Programm. Je nach verwendeter *tar*-Version weichen einzelne Parameter aber ab. Beispielsweise kann aus den Packages das kompatible Programm GNU-tar installiert werden, das anschließend mit *gtar* aufgerufen wird. Hier ist der Parameter für das Komprimieren beziehungsweise Dekomprimieren von bzip2-gepackten Dateien *-j*. *gtar* besitzt keine Manpage, für die Anzeige der Hilfe muß es wie die meisten GNU-Utilities mit *gtar --help* aufgerufen werden

11.9 Prozeßverwaltung

11.9.1 Programm zu bestimmtem Zeitpunkt starten

at [-k -m] Zeit [Datum][Inkrement][Job]

Startet ein Programm oder Kommando zu einem späteren Zeitpunkt.

Parameter:

-k	Führt den Job in der Korn-Shell aus.
-m	Schickt nach Ausführung des Jobs eine Nachricht an den Benutzer.
Zeit	Die Zeitangabe hat das Format »HHmm« (Stunde, Minute), wobei neben der 24-Stundennotation wahlfrei auch *am* und *pm* verwendet kann. Besondere Zeitangaben, die das System erkennt, sind *noon* und *midnight*.
Datum	Die Datumsangabe hat das Format Monatsname und Tageswert. Besondere Datumswerte sind *today* und *tomorrow*.
Inkrement	Hat das Format »+Zahl« oder den Begriff »next«, gefolgt von einer Angabe für Minute(n), Stunde(n), Tag(e), Woche(n), Monat(e) oder Jahr(e).

Beispiele:

```
at 7:30 pm June 30 t_backup
```

Führt das Kommando *t_backup* am 30. Juni um 19:30 aus.

```
at now next week t_backup
```

Führt das Kommando *t_backup* zur aktuellen Zeit in der nächsten Woche aus.

11.9.2 Mit »at« gestartete Jobs anzeigen

at *{-l | -r [Job]}*

Zeigt alle mit *at* gestarteten Jobs an beziehungsweise löscht entsprechende Jobs aus der Ausführungsliste.

Parameter:

-l	Listet alle zur Ausführung vorgesehen Jobs auf.
-r	Löscht Jobs aus der Ausführungsliste.

Beispiel:

```
at -r t_backup
```

Löscht den Job *t_backup* aus der Ausführungsliste.

11.9.3 Prozeß terminieren

kill *[-Signal] PID...*

Sendet ein Signal an die angegebenen Prozesse. Wird kein Signal angegeben, wird das Signal »15 TERM« verwendet.

Parameter:

-Signal		Die wichtigsten Signale sind
2	*INT*	Für »Interrupt«, unterbricht die Ausführung des Prozesses, wird bei Vordergrundprozessen durch [Strg]-[C] abgesetzt.
3	*QUIT*	Wird verwendet, wenn das Interrupt-Signal vom Prozeß ignoriert wird. Wird bei Vordergrundprozessen durch [Strg]-[\] abgesetzt.

9	*KILL*	Bricht den Prozeß ab, dieses Signal kann vom betroffenen Prozeß nicht ignoriert werden.
15	*TERM*	Terminiert den Prozeß.
23	*STOP*	Unterbricht den Prozeß, er kann allerdings mit dem Signal 25 weiter ausgeführt werden.
25	*CONT*	Für »Continue«, startet gestoppte Prozesse wieder.

Beispiel:

```
kill -9 201
```

Bricht den Prozeß mit der PID 201 gewaltsam ab.

11.9.4 Prozeßstatus anzeigen

***ps** [-acCehjKlmrSTuvwx] -M core -N system -= format -p PID - [-t terminal] \ [-U Benutzerliste] [W swap]*

Zeigt den aktuellen Status von Prozessen an.

Hier nur eine Auswahl der wichtigsten Parameter:

-a	Zeigt alle Prozesse an, auch die anderer Anwender.
-e	Zeigt alle Prozesse bis auf Kernel-Prozesse an.
-t terminal	Nur die Prozesse, die auf dem angegeben Terminal laufen, werden angezeigt.
-l	Generiert eine Liste mit allen Informationen.
-u	Zeigt alle dem aktuellen Benutzer zugehörigen Prozesse an.
-U Benutzerliste	Zeigt alle Prozesse der in der Benutzerliste aufgeführten und durch Kommata getrennten Benutzer an.

Beispiele:

```
ps
```

Zeigt alle Prozesse, die auf dem aktuellen Terminal laufen, an.

```
ps -l -u myuser,root
```

Zeigt alle Prozesse der Benutzer »myuser« und »root« an.

11.9.5 Prozeßübersicht

top [-SbiInquv] [-dzahl] [-szeit] [-ofeld] [-UBenutzer] [nummer]

Zeigt den aktuellen Status der Top-Prozesse im Vollbildmodus an und aktualisiert sie solange, bis das Programm mit [q] beendet wird.

Parameter:

-S	Anzeige von Systemprozessen wie Pager und Swapper ausblenden.
-b	Batch-Mode. Alle Eingaben am Terminal außer den Abbruchtasten werden ignoriert. Standardmodus auf dummen Terminals.
-i	Interaktiver Modus auf normalen »intelligenten« Terminals. Eingaben werden verarbeitet.
-I	Im Idle-Zustand befindliche Prozesse nicht anzeigen.
-n	Nicht-interaktiver Modus. Identisch mit -b.
-q	Ändern der Priorität von *top* (»renice«) zur Beschleunigung auf den Wert -20. Diese Option kann nur von root angegeben werden.
-u	UIDs nicht auf die Usernamen umsetzen.
-v	Ausgabe der Versionsnummer nach stdout und beenden.
-dzahl	Nur diese Zahl von Prozessen anzeigen und beenden.
-szeit	Angabe der Zahl von Sekunden zwischen zwei Updates des Bildschirms. Die Voreinstellung ist 2.
-ofeld	Sortierung der Anzeige nach dem hier angegebenen Feld. Erlaubt sind die Angaben (in Kleinbuchstaben) *cpu*, *size*, *res* und *time*.
-UBenutzer	Nur die Prozesse des hier angegebenen Benutzers anzeigen. Es muß der Name angegeben sein, die Angabe der UID funktioniert nicht.

top kennt einige Tastendrücke als Steuerbefehle, die während des Ablaufs direkt eingegeben werden können:

[Strg][L]	Bildschirm neu zeichnen.
[h] / [?]	Hilfebildschirm einblenden.
[q]	Programm beenden.
[d]	Anzahl der noch anzuzeigenden Bildschirme eingeben. Normalerweise arbeitet des Programm unbegrenzt beziehungsweise bis zum Beenden mit [q].
[n] / [#]	Zahl der anzuzeigenden Prozesse ändern.
[s]	Wartezeit zwischen zwei Aktualisierungen einstellen.
[S]	Umschalten zwischen dem Anzeigen und Verbergen der Systemprozesse.
[k]	Ein Kill-Signal an eine Liste von Prozessen schicken.
[r]	Ändern der Priorität einer Liste von Prozessen. Dies entspricht dem Aufruf von *renice*.
[u]	Anzeigen der Prozesse nur für einen bestimmten User. Nach dem Usernamen wird anschließend gefragt. Wird hier [+] eingegeben werden wieder alle User angezeigt.
[o]	Ändern der Sortierreihenfolge.
[e]	Anzeigen der Liste der Fehlermeldungen nach einem *kill* oder *renice*.
[i] / [I]	Umschalten der Anzeige der Idle-Prozesse.

```
load averages:  0.20,  0.48,  0.54                                  15:21:36
51 processes:  1 runnable, 49 sleeping, 1 on processor
CPU states:  9.0% user,  0.0% nice,  2.0% system,  0.5% interrupt, 88.6% idle
Memory: 50M Act, 5644K Inact, 316K Wired, 20M Exec, 16M File, 18M Free
Swap: 962M Total, 40M Used, 923M Free

  PID USERNAME PRI NICE   SIZE    RES STATE      TIME   WCPU    CPU COMMAND
  348 root       2  -17  1148K 2932K select    15:55  8.01%  8.01% artsd
  337 root       2    0   876K 7624K select     5:08  1.03%  1.03% kdeinit
  211 root       2    0  7772K   10M select     9:50  0.00%  0.00% XF86_Mach64
  358 root       2    0  2252K   11M select     1:15  0.00%  0.00% kdeinit
  364 root       2    0  1136K 8900K select     0:57  0.00%  0.00% kdeinit
  356 root       2    0  1444K   10M select     0:37  0.00%  0.00% kdeinit
    6 root      18    0    0K   20M syncer     0:27  0.00%  0.00% [ioflush]
  354 root       2    0  1348K 7684K select     0:24  0.00%  0.00% kdeinit
  361 root       2    0   872K 8264K select     0:19  0.00%  0.00% kdeinit
  766 root      10    0   312K 1116K nanoslee    0:05  0.00%  0.00% xosview
  807 root       2    0  1324K   12M select     0:04  0.00%  0.00% kdeinit
  350 root       2    0   860K 5268K select     0:02  0.00%  0.00% kdeinit
  371 root       2    0   840K 2896K select     0:02  0.00%  0.00% kalarmd
  353 root       2    0   852K 3896K select     0:01  0.00%  0.00% kdeinit
  370 root       2    0   796K 3052K select     0:01  0.00%  0.00% korgac
  331 root       2    0   248K 2968K select     0:01  0.00%  0.00% kdeinit
  178 root       2    0  2928K 1008K select     0:01  0.00%  0.00% perl
```

Abb. 11.2: top wird ausgeführt

11.10 Für den Administrator

11.10.1 Userdaten ändern

***chpass** [-a list] [-s newshell] [user]*

Mit dem Programm *chpass* wird die Benutzerdatenbank geändert. Es werden dabei jeweils nur die Daten angezeigt, die auch geändert werden dürfen. Dem Programm *chpass* können die zu ändernden Daten entweder auf der Kommandozeile angegeben werden oder alternativ werden sie im Systemeditor zum Editieren angeboten und nach dem Speichern der Pseudodatei aktualisiert.

-a	Ändern aller Anwenderdaten. Die Angabe auf der Kommandozeile erfolgt in dem Format, wie sie auch von *passwd* verlangt werden. Diesen Parameter kann nur der Systemadministrator verwenden.
-s newshell	Wechsel der Shell des Users auf die neue Shell.

Die Angabe der Parameter auf der Kommandozeile findet üblicherweise in Skripten Verwendung. Bequemer ist natürlich die Eingabe in einem Fullscreen-Editor.

Beispiel:

```
chpass -s /usr/pkg/bin/tcsh
chpass: updating the database...
chpass: done
```

Im interaktiven Modus ändert der Administrator die folgenden Werte beliebiger User:

Login	Der Login-Name des Users.
Uid	Die Login-ID des Users.
Gid	Die Login-Gruppe des Users.
Class	Die allgemeine Klassifikation des Users.
Change	Zeitpunkt, zu dem das User das Paßwort ändern muß.
Expire	Das Ablaufdatum des Accounts.
Full Name	Der echte Name des Users.
Office Location	Der Arbeitsplatz des Users.
Office Phone	Die Telefonnumer des Users.
Home Phone	Privattelefonnumer.
Other information	Diverse andere Informationen.

Der Anwender selbst kann nur die folgenden Werte anpassen: »Shell«, »Full Name«, »Office Location«, »Office Phone«, »Home Phone« und »Other information«.

Beim Wechsel der Shell kann nur ein Programm angegeben werden, das in der Datei »/etc/shells« eingetragen ist. Auf einem System mit einigen installierten Shells sieht diese Datei beispielsweise so aus:

```
#       $NetBSD: shells,v 1.3 1996/12/29 03:23:07 mrg Exp $
#
# List of acceptable shells for chpass(1).
# Ftpd will not allow users to connect who are not using
# one of these shells.

/bin/sh
/bin/csh
/bin/ksh
/usr/pkg/bin/bash
/usr/pkg/bin/tcsh
/usr/pkg/bin/zsh
```

Die eigentlich zum System gehörenden Shells befinden sich in »/usr/bin«, zusätzlich installierte Shells in »/usr/local/bin«. Die im Kompatibilitätsverzeichnis zu Linux eventuell installierte »/usr/compat/linux/bin/bash« ist hier nicht aufgeführt. Sie wird nur von Linux-Programmen verwendet. Soll sich ein im System definierter User (bespielsweise weil es sich um einen reinen Netzwerkuser handelt) nicht einloggen dürfen ist als zusätzliche Shell das Programm »/sbin/nologin« in die Definitionsdatei einzutragen und dann dieses Programm für den entsprechenden User als Login-Shell anzugeben.

11.10.2 Anwender hinzufügen

useradd *-D [-b basedir] [-e expiry] [-f inactive] [-g group]*
[-r lowuid..highuid] [-s shell] [-L class]

oder

useradd *[-G group] [-b basedir] [-c comment] [-d homedir] [-e expiry]*
[-f inactive] [-g group] [-k skeletondir] [-m] [-o] [-p password]
[-r lowuid..highuid] [-s shell]
[-u uid] [-v] user

Fügt einen neuen Anwender hinzu. Dazu müssen alle Daten auf der Kommandozeile beim Aufruf angegeben werden. Ein Aufruf für das Einrichten des Users *user* sieht so aus:

```
useradd -G wheel -b /home -d /home/user -g wheel -k /etc/skel -m -s \
/bin/csh -u 1005 user
```

Wichtig ist die Angabe von *-m*, wenn das Verzeichnis nicht bereits angelegt ist. Die wichtigsten Optionen im einzelnen:

[-G group]	Sekundärgruppe. Wird in die »/etc/group« eingetragen.
[-b basedir]	Basisverzeichnis, normalerweise »/home«.
[-d homedir]	Der komplette Pfad zum Heimatverzeichnis.
[-g group]	Primärgruppe. Wird in die Datei /etc/group eingetragen
[-k skeletondir]	Das Verzeichnis in dem sich die Templates für die Konfigurationsdateien befinden. Normalerweise ist das »/etc/skel«.
[-m]	Ist das Verzeichnis noch nicht vorhanden wird es neu angelegt. Wichtig: sonst nicht!
[-p password]	Angabe des **verschlüsselten** Paßworts. Normalerweise nicht festsetzen, sondern anschließend mit *passwd* anlegen!
[-r lowuid..highuid]	Bereich für die User-ID. Besser direkt bestimmen.
[-s shell]	Die zu verwendende Shell. Die angegebene Shell muß in der Datei »/etc/shells« eingetragen sein.
[-u uid]	Die zu verwendende numerische User-ID
[-v]	Verbose Mode
user	Der Name des neuen Users im System

11.10.3 Datenträger anmelden

***mount** [all] | [-t Typ] [-o Mountoptionen] [Device-Knoten] [Mountpunkt]*

Macht die Inhalte eines Datenträgers als Dateisystem verfügbar. Das Programm kennt zahlreiche Parameter und Optionen von denen hier nur wichtigsten wiedergegeben werden. Lesen Sie die Spezialparameter die Manpage des Programms nach. Parameter:

all	Hängt alle Standard-Dateisysteme an.
-t Typ	Angabe des Typs des Dateisystems, beispielsweise ffs, cd9660, ext2fs, msdos, union, ntfs und so weiter.
-o Optionen	Beim Einbinden zu berücksichtigende Optionen, zum Beispiel *rdonly* wenn der Datenträger schreibgeschützt eingebunden werden soll.

Beispiel:

```
mount -t cd9660 /dev/cd0d /mnt/cdrom
```

Hängt den Geräteknoten »/dev/cd0d« unter das Verzeichnis » /mnt/cdrom« ein.

Unter BSD gibt es einige weitere Mount-Anweisungen, die auf jeweils ein Dateisystem spezialisiert sind.

mount_ffs | mount_ufs device mountpunkt	Hängt ein natives BSD-Dateisystem ein.
mount_cd9660 device mountpunkt	Hängt ein CD-Laufwerk mit ISO9660-konformem Dateisystem an. Lange Dateinamen nach RockRidge und Joliet werden unterstützt.
mount_ext2fs device mountpunkt	Hängt ein Linux-Dateisystem an (Ext2 oder Ext3).
mount_msdos device mountpunkt	Hängt ein MS-DOS-Dateisystem an.
mount_nfs device mountpunkt	Hängt ein NFS-Dateisystem an.
mount_ntfs device mountpunkt	Hängt ein NTFS-Dateisystem ein. Unterstützt werden NTFS von Windows NT und NFTS5 von Windows 2000.
mount_union [-br -o -r] dir1 uniondir	Verbindet zwei Verzeichnisse zu einer Einheit (Vorsicht! Das Union-System ist nicht ausgereift!).
mount_lfs [-bdns] [-N nsegs] [-o options] device mountpunkt	Bindet ein Laufwerk, das im NetBSD-eigenen Log-Structured Filesystem angelegt wurde, ein.

Die weiteren Mount-Programme *mount_fdesc*, *mount_ados*, *mount_mfs*, *mount_null*, *mount_filecore*, *mount_overlay*, *mount_umap*, *mount_portal* und *mount_procfs* werden normalerweise nicht benötigt. Falls Sie auf ein AmigaDOS-Dateisystem (*mount_ados*), ein Filecore-Dateisystem von RISC OS (*mount_filecore*) oder eines der experimentellen Dateisysteme wie NullFS, Overlay oder Portal stoßen, sollten Sie die Manpages des Mount-Programms konsultieren. Das HFS-Dateisystem des Macintosh werden Sie hier vergeblich suchen. Es ist auch bei den Macintosh-Ports nicht integriert. Es gibt aber einige HFS-Tools in den Packages. Im Gegensatz zu FreeBSD gibt es auch kein *mount_smbfs* für Samba-Mounts.

Hier muß man sich unter NetBSD mit dem Userland-Programm *Sharity Light* aus den Packages (»net/sharity-light«) behelfen.

Bei allen Mounts gilt, daß *device* im Verzeichnis »/dev« vorhanden sein muß. Der Mountpunkt muß ein Pfad auf ein vorhandenes Verzeichnis sein. Dieses Verzeichnis muß übrigens nicht leer sein. Ein eventuell vorhandener Inhalt wird aber (außer bei Union-Mounts) ausgeblendet.

Beispiel:

```
mount_cd9660 /dev/cd0d /mnt/cdrom
```

Dieser Befehl hängt unter NetBSD ein ATAPI- oder SCSI-CD-Laufwerk in das Verzeichnis »/mnt/cdrom«.

Die Einzelmountbefehle befinden sich wie *mount* selbst im Verzeichnis »/sbin«. Mit der Installation neuer Dateisysteme werden automatisch zusätzliche *mount*-Befehle zugefügt. Wird beispielsweise aus den Ports das Samba-Dateisystem *smbfs* installiert, befindet sich anschließend in »/sbin« ein zusätzlicher Befehl *mount_smbfs*.

Die Angabe des Quellgeräts darf entfallen, wenn die Definition des Mountpunkts in der Datei »/etc/fstab« eingetragen ist. Hier eine typische »fstab«-Datei von einem System mit zwei IDE-Platten, einem CD-ROM-Laufwerken und einem Diskettenlaufwerk im Format 1,44 Mbyte:

```
/dev/wd1a       /           ffs     rw              1 1
/dev/wd1b       none        swap    sw              0 0
/dev/wd0b       none        swap    sw              0 0
/dev/wd2b       none        swap    sw              0 0
/dev/wd1e       /usr        ffs     rw              1 2
/dev/wd2e       /pub        ffs     rw              1 2
kernfs          /kern       kernfs  rw              0 0
procfs          /proc       procfs  rw              0 0
/dev/cd0d       /mnt/cdrom  cd9660  ro,noauto       0 0
/dev/wd2h       /mnt/dos    msdos   rw              0 0
/dev/fd0a       /mnt/floppy ffs     rw,noauto       0 0
/dev/fd0a       /mnt/A:     msdos   rw,noauto       0 0
```

Wenn hier das CD-Laufwerk eingebunden werden soll, reicht die Angabe von

```
mount /mnt/cdrom
```

Die restlichen Angaben holt sich das System aus der »/etc/fstab«. Im Gegensatz zu Linux wird bei NetBSD übrigens nicht zwischen FAT- und VFAT-Partitionen unterschieden, so daß der Parameter *msdos* für beides gilt. Der Name für CD-ROM-Laufwerke unterscheidet sich auch von Linux, bei dem das Format *iso9660* anzugeben ist.

NetBSD unterstützt über einen Umweg auch das Mounten von ISO-Images. Diese ISO-Images sind normalerweise Diskettenimages oder Rohdaten-Images von CD-ROMs. Mit dem folgenden kleinen Skript wird ein anzugebendes ISO-CD-Image in ein Verzeichnis »/mnt/iso« (das vorhanden sein muß) eingebunden. Auf das Image kann dann genauso wie auf ein reguläres CD-Laufwerk zugegriffen werden:

```
#!/bin/sh
# mount_iso
# Mount an ISO image in NetBSD

if [ "$1" != "" ] ; then
  if [ ! -f $1 ] ; then
    echo ISO-Image not found
    echo ISO mount aborted"!"
  else
    vnconfig /dev/vnd0d $1
    mount -t cd9660 /dev/vnd0d /mnt/iso
  fi
fi
```

Dabei wird ein virtuelle Gerät mit dem Befehl *vnconfig* angelegt und über dieses dann das Dateisystem nach »/mnt/iso« eingebunden. Die Vorgehensweise unterscheidet sich drastisch von der entsprechenden bei Linux.

ISO-Images müssen genauso wie normale Mounts auch wieder mit einem unmount (Befehl *umount*) aus dem System ausgebunden werden. Das entsprechende Skript dazu sähe folgendermaßen aus:

```
#!/bin/sh
# umount ISO image
umount /mnt/iso
vnconfig -u /dev/vnd0d
```

Hierbei wird der Mountpunkt gelöst und der virtuelle Knoten wieder freigegeben. NetBSD 1.6 unterstützt mit dem generischen Kernel vier virtuelle Geräte »vnd0« bis »vnd3«. Allerdings sind die Geräte 3 und 4 in »/dev« normalerweise nicht definiert und müssen mit dem MAKEDEV-Skript erst angelegt werden.

11.10.4 System beenden

***shutdown** [-d -F -h -i -p -k -p -r][+Zeitnachricht]*

Beendet den Systembetrieb durch Herunterfahren des Unix-Rechners. Das Kommando kann nur von *root* oder entsprechend berechtigten Benutzern verwendet werden. Die Parameter:

-d	Erzeugt einen Dump.
-h	Hält das System vollständig an.
-f	Fastboot. Es wird kein Dateisystem-Check durchgeführt.
-k	Wirft alle Benutzer aus dem System. Kein Systemabschluß
-r	Startet nach erfolgten Systemabschluß das Unix-System erneut.
-p	Führt *poweroff* anstelle von *halt* durch. Ein ATX-System wird damit komplett abgeschaltet.
+Zeitnachricht	Im Format Anzahl Minuten bis zum Systemabschluß oder Stunde:Minute für den Zeitpunkt des Systemabschlusses. Der Standardwert ist eine Minute.

Beispiel:

```
shutdown -p now
```

hält das System sofort an. Ein ATX-System wird damit komplett ausgeschaltet. Wird der Befehl mit dem Parameter *-p* auf einem System aufgerufen, das das Ausschalten nicht unterstützt, wird nur bis zur Ausschaltmeldung äquivalent zu *shutdown -h* heruntergefahren.

```
shutdown -r now
```

startet das System sofort neu.

NetBSD enthält außer dem generischen Befehl auch noch die Programme *halt* und *reboot*, mit denen das System ebenfalls angehalten beziehungsweise neu gestartet werden kann. Mit *poweroff* wird ein ATX-Board heruntergefahren und ausgeschaltet. Das funktioniert nicht nur auf der Intel-Plattform, sondern auf allen Maschinen, die das Ausschalten per Software zulassen, also beispielsweise auch auf Macintoshs.

11.10.5 Anwender entfernen

***userdel** [username]*

Entfernt einen Anwender aus dem System. Das Programm fragt nach, ob das Home-Verzeichnis ebenfalls gelöscht werden soll.

11.10.6 Dateisystem abhängen

***umount** [-f] {-a|all|-n Knoten|-t Typ|Datei|Verzeichnis|Dateisystem}*

Hängt ein zuvor angehängtes Dateisystem wieder ab.

Parameter:

-f	Erzwingt das Abhängen eines entfernten Dateisystems.
-a	Hängt alle angehängten Dateisysteme wieder ab.
all	Hängt alle Dateisysteme wieder ab, mit Ausnahme der Dateisysteme, die in der Datei »/etc/filesystems« mit dem Attribut *mount=automatic* verzeichnet sind.
-n Knoten	Hängt alle Verzeichnisse vom entfernten System-Knoten wieder ab.
-t Typ	Hängt alle Verzeichnisse des Typs »Typ« wieder ab.

Beispiel:

```
umount /mnt
```

Entfernt das Gerät, das vorher unter »/mnt« gemounted wurde, wieder aus dem System.

11.11 Interessante Tools

Unix enthält von Hause aus eine Vielzahl von speziellen, extrem mächtigen Werkzeugen. Leider sind diese nicht gerade bequem zu bedienen. Wer sich mit diesen Tools auskennt, kann aber oftmals Problemlösungen automatisieren, die sonst sehr viel Handarbeit bedeuten würden. Inzwischen gibt es diese Tools auch für andere Betriebssysteme wie MS-DOS und OS/2. Die hier aufgeführten Beispiele awk und vi sollen ein Anreiz für den Leser sein, sich mit diesen Tools etwas näher zu beschäftigen. Sie werden sehen – es lohnt sich!

11.11.1 awk

Dieses Tool dient zur umfangreicheren Textmanipulation. Es liest eine Textdatei und erstellt gemäß seiner Arbeitsanweisung eine Ausgabe. Diese Arbeitsanweisung kann entweder in den Aufrufparametern oder in einer separaten Beschreibungsdatei überge-

ben werden. Als Grundlage für die Umformung dienen dabei Felder (Spalten) und Records (Zeilen) innerhalb der Quelldatei, die durch frei wählbare Separatoren definiert werden. Durch die weitere Analyse der Eingabe anhand von Suchmustern können neben den Feldern auch andere Textstücke zur »Triggerung« von Kommandos zum Einsatz kommen.

Einige wichtige Variablen sind:

FS	Feldseparator zur Trennung der Felder der Eingabedatei. Voreingestellt ist das Leerzeichen.
OFS	Ausgabe-Feldseparator. Voreingestellt ist das Leerzeichen.
RS	Separator zur Trennung von Records in der Eingabedatei. Voreingestellt ist Newline.
ORS	Separator zur Trennung von Records in der Eingabedatei. Voreingestellt ist Newline.
NR	Aktuelle Zeilennummer.
NF	Anzahl der Felder im aktuellen Record.
$i	i-tes Input-Feld des aktuellen Records.
$0	Aktuelle Zeile.

Mit solchen Variablen können die Kommandos definiert werden.

Beispiel:

```
awk '{print $5,$2,$1}' myfile
```

Die Datei »myfile« wird durchsucht und die Reihenfolge der Spalten wird verändert. Die Ausgabe erfolgt auf den Bildschirm.

```
awk '/UNIX/{print $0}' myfile
```

Es werden alle Zeilen in der Datei »myfile« aufgelistet, in denen der Suchbegriff »UNIX« zu finden ist.

```
awk -F: 'BEGIN {OFS=","}{print $2,$1}' myfile
```

Der Eingabefeldseparator wird auf Doppelpunkt »:« gesetzt und der Ausgabefeldseparator auf Komma »,«. Dann werden die ersten und zweiten Spalten aller Zeilen vertauscht. Aus »1:2« wird also »2,1«.

11.11.2 vi

vi [+n] Dateiname

Der Visual Editor vi ist ein recht gebräuchlicher Editor. Gegenüber anderen Editoren hat er den Vorteil, daß er zum Grundsystem gehört und nicht erst nachträglich installiert werden muß. Dieser Editor unterscheidet zwischen Kommandomodus und Eingabemodus. Nach dem Aufruf ist der vi zunächst einmal im Kommandomodus. Hier können nur Befehle – also kein Text – eingegeben werden.

Die wichtigsten Kommandos zur Dateimanipulation sind:

:w	write file, die Datei wird gespeichert.
:q	quit, der Editor wird verlassen.
:wq	write and quit, die Datei wird geschrieben und der Editor verlassen.
:q!	quit, der Editor wird verlassen und eventuelle, nicht gespeicherte Änderungen werden verworfen.
:e name	edit file, eine andere Datei wird geladen.
:w name	write file name, die Datei wird unter anderem Namen gespeichert.
:w! name	Eine eventuell vorhandene Datei gleichen Namens wird überschrieben.
:sh	Shell, ruft die Kommandozeile auf und kehrt nach dem Beenden dieser wieder zum vi zurück.
:! command	Führt den Befehl *command* aus und kehrt nach dessen Beendigung wieder zum vi zurück.
:x	Exit, beendet den vi.
:x!	Exit, beendet den vi ohne eventuelle Nachfrage.

Im Kommandomodus können Sie die Datei in aller Ruhe ansehen. Sobald Sie dann den Cursor an die richtige Stelle gebracht haben, können Sie den Text unter anderem mit den folgenden Kommandos editieren.

a	Text hinter dem Cursor einfügen.
i	Text vor dem Cursor einfügen.
A	Text am Ende der Zeile einfügen.
I	Text am Anfang der Zeile einfügen.
o	Zeile unter dem Cursor einfügen.

O	Zeile über dem Cursor einfügen.
rx	Ersetze einzelnen Buchstaben durch »x«.
R	Text überschreiben.
X	Zeichen vor der Cursorposition löschen.
D	Rest der Zeile löschen.
C	Rest der Zeile ändern.

Mit Druck auf die [Esc]-Taste gelangen Sie zurück in den Kommandomodus.

11.12 Nützliche Zusatzpakete

Eine Spezialität von BSD sind die sogenannten Packages. Neben grundlegenden Dingen wie beispielsweise der Installation eines WindowManagers finden sich dort zahlreiche sehr nützliche Werkzeuge, die zwar nicht zum Standardumfang einer Systeminstallation gehören, aber dennoch sehr empfehlenswert sind. Je nach Anwendungszweck kann so ein bestimmtes System zugeschnitten werden. Einige dieser Werkzeuge verdienen es, an dieser Stelle erwähnt zu werden.

11.12.1 Super Kill

***skill** [-Signal] [-i -v -w -n] Prozeß*

Dieses Kommando dient ebenso wie das systemeigene *kill* zum Senden von Signalen an Prozesse. Im Gegensatz zu diesem ist es aber nicht nötig, die Prozeßnummer anzugeben. Das umständliche Suchen mittels *ps* und *grep* gehört damit der Vergangenheit an. Achtung! Der Benutzer *root* kann auf die Prozesse aller anderen Benutzer zugreifen. Wenn *root* ein *skill* durchführt, werden daher die entsprechenden Prozesse aller Benutzer terminiert.

-i	Interaktiv: fragt bei jedem Prozeß nach, ob er wirklich das Signal erhalten soll.
-v	Gibt die Prozeßnummern der beeinflußten Prozesse aus.
-w	Gibt eine Warnung aus, wenn ein bestimmter Prozeß nicht erreicht werden kann.
-n	Gibt nur die Prozeßnummern aus, sendet aber kein Signal.

Beispiel:

```
skill -9 xclock
```

Bricht alle laufenden xclocks ab, die vom entsprechenden Benutzer terminiert werden dürfen.

***snice** [+-Priorität] [-i -v -w -n] Prozeß*

Ebenfalls im *skill*-Paket enthalten, dient dieses Kommando zur Änderung der Priorität einzelner Prozesse. Es entspricht dem Systemkommando *nice*. Grundsätzlich hat nur *root* das Recht, die Priorität eines Prozesses zu erhöhen. Alle anderen Benutzer können nur jeweils niedrigere Werte angeben. Daher dürfte dieser Befehl das wohl am wenigsten verwendete aller Unix-Kommandos sein.

Für Verwirrung sorgt die Tatsache, daß eine negative Angabe die Priorität erhöht und eine positive Angabe sie verringert.

-i	Interaktiv: fragt bei jedem Prozeß nach, ob er wirklich beeinflußt werden soll.
-v	Gibt die Prozeßnummern der beeinflußten Prozesse aus.
-w	Gibt eine Warnung aus, wenn ein bestimmter Prozeß nicht erreicht werden kann.
-n	Gibt nur die Prozeßnummern aus, ändert aber nicht die Priorität der entsprechenden Prozesse.

Beispiel:

```
snice +10 xclock
```

Verringert die Priorität von xclock um 10.

11.12.2 Administrative Gruppen einrichten

***sudo** [befehl]*

Generell sollte schon aus Sicherheitsgründen ein Arbeiten unter dem *root*-Account vermieden werden. Auch ein Setzen des SUID-Bits hat sich in der Vergangenheit als sehr sicherheitskritisch herausgestellt. Um dennoch bestimmten Personen oder Gruppen einen Zugang zu kritischen Kommandos wie etwa *mount* oder *shutdown* zu gewähren, kann das *sudo*-Paket eingesetzt werden. Die Rechte werden in einer Textdatei namens »/usr/pkg/etc/sudoers« einzeln vergeben.

11.12.3 Neuen angepaßten Kernel bauen

adjustkernel *[-o{utfile} new-config] [-m{esg} dmesg-file] [-r{emove}] [-d{ebug}] -f{ile} source-config*

Dieses Programm, das aus dem Package »/usr/sysutil/adjustkernel« generiert wird, erzeugt eine neue angepaßte Kernel-Konfigurationsdatei aus dem aktuellen Kernel und den beim Booten gefundenen Geräten.

Mit dem Befehl

```
adjustkernel -f GENERIC -o MYKERNEL
```

wird die neue Kernel-Konfigurationsdatei »MYKERNEL« aus dem gebooteten generischen Kernel erzeugt. Nach dem Kopieren in das Konfigurationsverzeichnis wird damit auf sehr bequeme Weise mit

```
config MYKERNEL
cd ../compile/MYKERNEL
make depend && make
cp /netbsd /onetbsd
cp netbsd /
shutdown -r now
```

ein neuer Kernel erzeugt, installiert und gestartet.

12 X Window

von Wolfgang Soltendick

Unix ist vom Grundsatz her ein textorientiertes Betriebssystem. Es gibt zwar zaghafte Versuche, im Textmodus Grafik einzusetzen, die Ergebnisse erlangten aber nie auch nur marginale Bedeutung.

Parallel wurde eine grafische Oberfläche »X« entworfen, die in den kommerziellen Unix-Systemen Einzug hielt. X liegt inzwischen in der Version 11 vor, diese im Release 6. Daraus leitet sich der Name X11 R6 ab. Einige Unix-Systeme wie SCO OpenServer verwenden noch die ältere Version X11 R5.

Parallel zu X11 wurde die freie Variante XFree86 entwickelt, die bei fast allen freien Unix-Systemen verwendet wird. Bis zum Jahr 2000 gab es allein die Version XFree86 Version 3, die sich im Verzeichnisaufbau an X11 Release 6 orientierte. Inzwischen ist parallel die Version 4 von XFree86 verfügbar. Sie ist zwar bei einigen Linux- und BSD-Systemen die Standard-Oberfläche, besitzt aber den Nachteil, daß nur Treiber für neue Grafikkarten enthalten sind, während XFree86 3 auch besondere Beschleuniger für Standard-Grafikkarten enthält.

Während in den NetBSD-Versionen bis 1.5.2 standardmäßig die Version XFree86 3 eingerichtet wurde, hat sich das ab NetBSD 1.5.3 grundlegend geändert. Hier wird die Version 4 installiert, was bedeutet, daß neuere Karten gut unterstützt werden. Bei älteren Karten dagegen besteht aber das Problem, daß sie mitunter kein vernünftiges Bild mehr liefern, so daß man in einem solchen Fall manuell eingreifen muß. Das bedeutet, daß man den entsprechenden Treiber von Hand in das System einbinden muß, damit man das X-Window-System nutzen kann.

12.1 Installation

Der einfachste Weg, die Grundbestandteile von X Window zu installieren, ist die Auswahl der entsprechenden Option *Standard mit X* gleich bei der Einrichtung des Betriebssystems. Damit werden beispielsweise alle X-Server auf die Platte kopiert, die Schriften und einige grundlegende Programme. Insgesamt werden die folgenden Pakete installiert, die sich im Verzeichnis »i386/binary/sets« befinden.

Hat man nur das Standardsystem installiert, lassen sich diese Pakete einfach nachinstallieren. Dazu bindet man das CD-ROM-Laufwerk ein, wechselt in das Verzeichnis »i386/binary/sets« und richtet die Pakete mit der folgenden Sequenz ein:

```
# ( cd / ; tar -zxpf - ) < NAME.tgz
```

Dabei bezeichnet *NAME* den Namen der einzurichtenden Datei. Da in diesen Paketen keine Abhängigkeiten gespeichert sind, ist die Reihenfolge der Installation beliebig. Man sollte nur sicherstellen, daß man auch wirklich alle einrichtet. Außerdem ist es wichtig, daß sie aus dem Wurzelverzeichnis »/« installiert werden, da sie relative Pfade enthalten. In der oben beschriebenen Kommandozeilensequenz wird das durch ein

```
# cd /
```

am Anfang erledigt.

Paket	Größe (komprimiert/unkomprimiert) in Mbyte	Beschreibung
xbase.tgz	4,1 / 11,3	Das Paket enthält die Basisdateien für eine Client-X-Umgebung, mit Ausnahme der X-Server.
xcomp.tgz	3,3 /12,5	In diesem Paket befinden sich die Bibliotheken und Include-Dateien, die zur Übersetzung von X-Programmen benötigt werden.
xcontrib.tgz	0,3 / 0,8	Eine Reihe kleiner Programme für X, wie beispielsweise xedit, xcalc oder xman.
xfont.tgz	15,2 / 17,6	Die für das X-Window-System benötigten Schriften.
xmisc.tgz	0,02 / 0,05	SSH-Erweiterungen für OpenSSH.
xserver.tgz	7,3 / 19,5	Alle X-Server der Version 4. In der Regel benötigt man aber natürlich für eine Grafikkarte nur einen Server, so daß man die restlichen im Prinzip wieder manuell löschen kann.

Tabelle 12.1.: Die X-Basispakete auf der Intel-Plattform

12.2 Der Verzeichnisaufbau

Für Unix typisch ist die im Gegensatz beispielsweise zu Windows streng definierte Verzeichnisstruktur. Es gibt zwar unter Linux einige Abweichler, NetBSD hält sich aber streng an die Norm. Das X-Grundsystem befindet sich deshalb immer in »/usr/X11R6«. Dieses Verzeichnis enthält wie gewohnt mehrere Unterverzeichnisse, in denen die entsprechenden Dateien abgelegt sind (Tabelle 12.2).

Ganz kompliziert ist die Position der X-Konfigurationsdateien gelöst. Sie befinden sich nicht etwa – was ja logisch wäre – in »/usr/X11R6/etc« (oder in /etc beziehungsweise /etc/X11), sondern vielmehr in Unterverzeichnissen von »/usr/X11R6/lib/X11/«. Auf diese Dateien, die entscheidend für das Verhalten der grafischen Oberfläche sind, wird im Verlauf dieses Kapitels noch ausführlich eingegangen.

Verzeichnis	Inhalt
/usr/X11R6/bin	Alle Binärdateien
/usr/X11R6/etc	Alle X11-Konfigurationsdateien außer der XF86Config.
/usr/X11R6/include	C-Headerdateien der grafischen Oberfläche.
/usr/X11R6/info	GNU-Infodateien von GNU-X11-Programmen.
/usr/X11R6/lib	Alle X11-Bibliotheken.
/usr/X11R6/libexec	Normalerweise ist dieses Verzeichnis vorhanden, aber leer.
/usr/X11R6/man	Man-Pages zu den X-Programmen.
/usr/X11R6/sbin	GNU-Hilfsprogramme.
/usr/X11R6/share	Diverse Dateien der Zusatzpakete.

Tabelle 12.2: Der Verzeichnisaufbau

12.3 Konfiguration

Nachdem man die grundlegenden X-Dateien installiert hat, muß man das System konfigurieren. Dafür ruft man ein Programm auf, das nach einigen Fragen automatisch eine entsprechende Konfigurationsdatei schreibt, mit der dann das X-Window-System gestartet wird. Führt man diese Schritte nicht aus und versucht man das System zu starten, sieht man nur eine Reihe von Fehlermeldungen, weil beispielsweise nicht klar ist, welcher X-Server tatsächlich gestartet werden soll.

Die eigentliche Konfiguration kann man im folgenden bequem im Grafikmodus mit dem Programm *xf86cfg* oder unbequem im Textmodus ausführen. Im letzteren Fall startet man das Programm *xf86config* und beantwortet die entsprechenden Fragen.

12.3.1 xf86config

Das Konfigurationsprogramm *xf86config* wird bei den Standardpaketen der Distribution mitgeliefert. Man stellt damit die grundlegenden Daten der Konfigurationsdatei »/usr/X11R6/lib/X11/XF86Config« ein, allerdings weiß das Programm nichts über spezielle Einstellungen für bestimmte X-Server. Es ist rein textbasiert und das ist nicht gerade ein bequemer Weg für die Konfiguration.

Startet man das Programm, erscheint zuerst eine Sicherheitsabfrage, ob man tatsächlich die Konfiguration ausführen will. An dieser Stelle kommt man mit [Strg]+[C] wieder zurück, mit der Eingabetaste fährt man fort.

Man wird nach dem Typ der Maus beziehungsweise des entsprechenden Protokolls gefragt. Hier hat man mehrere Optionen, wobei diese aber weitgehend nur die grundlegenden Möglichkeiten wie serielle oder Busmaus bieten. Die explizite Angabe einer USB-Maus sucht man beispielsweise vergebens. Der Kernel von NetBSD ist so gut, daß man sich um die Schnittstelle im Normalfall überhaupt keine Gedanken machen muß.

Im Gegensatz zu anderen Betriebssystemen werden die Schnittstelle der Tastatur und er Maus über den wscons-Treiber gekapselt.

Deshalb sollte man bei der Maus immer »wsmouse« wählen.

Danach muß man die Gerätedatei angeben, über die die Maus angesprochen werden kann. Auch hier gilt bei NetBSD, daß »/dev/wsmouse«, eigentlich immer paßt, egal ob es sich um eine PS/2 oder eine USB-Maus handelt[1].

Durch Beantworten der nächsten Frage spezifiziert man, welchen Tastaturtyp man normalerweise einsetzt. Bei einer deutschen Tastatur (mit Windows-Tasten) kommt die Option 4 (*Generic 105-key (Intl) PC*) zum Einsatz, die genau die spezifizierten hundertundfünf Tasten besitzt. Anschließend legt man die Anordnung der Tasten fest, wozu die entsprechende Sprache angezeigt wird. Da die komplette Liste meist nicht auf eine Seite paßt, wird die Ausgabe nach achtzehn Einträgen erst einmal beendet und man kann entweder eine Zahl eingeben, um einen speziellen Eintrag auszuwählen, oder die Eingabetaste drücken, um den nächsten Teil der Liste abzurufen. Das macht man so lange, bis man die gewünschte Zeile gefunden hat. Für eine deutsche Tastatur sollte man die Nummer 27 nehmen, damit die Anordnung der Tasten stimmt.

Nun vergibt man einen Variant-Namen für die deutsche Anordnung der Tasten. In der Regel reicht es aber, wenn man den Default-Wert durch Drücken der Eingabetaste übernimmt. Die folgende Frage kann man leider nur mit [y] oder [n] beantworten; hier geht es um Erweiterungen für die Tastatur. Da man die Konfigurationsdatei später natürlich noch ändern kann, reicht es jetzt erst einmal, diese mit [n] zu beantworten.

Die sich nun anschließenden Fragen beziehen sich auf die Spezifikationen des Monitors. Hier muß man zuerst die horizontale Wiederholfrequenz (beziehungsweise einen passenden Bereich) aus einer Liste wählen, alternativ kann man auch einen Bereich direkt angeben. Zu beachten ist aber, daß bei einer falschen Wahl der Monitor unter Umständen Schaden nehmen kann, so daß man eher einen Monitor aussuchen sollten, der etwas unter den tatsächlichen technischen Daten liegt. Allerdings wird dadurch unter Umständen das Bild nicht mit der maximal möglichen Frequenz angezeigt. Wichtig bei der Wahl ist die obere Angabe des Frequenzbereichs, die man in der Dokumentation zum Monitor nachschlagen sollte. Die besseren neuen Monitore sollten beispielsweise mit maximal 95 kHz betrieben werden können, entsprechend Punkt 10 in der Liste.

Im nächsten Schritt wählt man die vertikale Synchronisationsfrequenz des Monitors. Für diese ist beispielsweise der Punkt 3 (50 bis 100 Hz) eine gute Wahl, wobei die tatsächliche Frequenz abhängig von der gewählten Auflösung ist. Besitzt man übrigens einen TFT-Flachbildschirm, kann man das System ruhig mit 50 Hz betreiben; man sieht dabei kein Flackern. Das liegt daran, daß die TFT-Monitore anders arbeiten als die Röhrenmonitore, so daß sie mit geringeren Frequenzen betrieben werden können, ohne daß man ein Flimmern sehen würde. Das ändert sich natürlich, wenn man mit diesen Einstellungen einen Röhrenmonitor ansprechen würde, denn in diesem Fall wäre ein Arbeiten fast unmöglich.

---

[1] Natürlich muß im Kernel das entsprechende Gerät auch definiert sein. Im GENERIC-Kernel ist das beispielsweise für PS/2- und USB-Schnittstelle der Fall.

Im nun folgenden Schritt muß man noch einen String angeben, der zur Identifikation des Monitors dient. Er wird nur intern verwendet, so daß auch die Angabe eines Buchstabens ausreicht.

Daraufhin schließt sich die Konfiguration der Grafikkarte an. Man hat dabei die Möglichkeit, eine Karte aus einer Datenbank zu wählen, wobei auch immer der darauf verwendete Chipsatz mit angezeigt wird. Allerdings muß man diese Suche explizit mit »y« einleiten. Zu beachten ist, daß man nur dann ein Modell aus der Datenbank auswählen sollte, wenn man sicher ist, daß es sich um die eigene Karte handelt. Sonst kann es zu nicht vorhersagbaren Effekten kommen, die meist bewirken, daß man kein Bild auf dem Bildschirm sehen kann.

Ist die eigene Karte nicht in der Datenbank enthalten, ist das nicht weiter tragisch, denn man kann immer noch eine allgemeine VGA-kompatible Karte wählen, die zu Anfang in der Liste steht. Man muß dann die entsprechende Nummer eingeben, um eine spezielle Karte auszuwählen, ansonsten blättert man sie mit der Eingabetaste durch.

Nach der Wahl werden die Einstellungen, die für diese spezielle Karte gelten, noch einmal angezeigt. Ist man mit diesen einverstanden, muß man nur die Eingabetaste drücken, um die Installation fortzusetzen. Ansonsten läßt sie sich mit [Strg]+[c] abbrechen.

Doch damit ist die Konfiguration noch nicht beendet, denn im nächsten Schritt ist anzugeben, wieviel Speicher auf der Grafikkarte vorhanden ist. ist man sich nicht sicher, schlägt man in der Dokumentation nach. Wie auch beim Monitor ist es weiterhin erforderlich, einen String zur Beschreibung der Grafikkarte anzugeben. Dieser wird zur Identifikation benutzt.

Ist das erledigt, gibt man an, welche Farbmodi bei welcher Auflösung verfügbar sein sollen. Sie hängen von der Angabe des Speichers auf der Grafikkarte ab, so daß besonders bei höheren Auflösungen Modi mit einer größeren Farbanzahl nicht mehr verfügbar sind, weswegen man die großen Auflösungen bei den höheren Farbtiefen möglichst nicht verwenden sollte.

Im nächsten Schritt kann man eine virtuelle Auflösung konfigurieren, die bewirkt, daß der Anzeigebereich größer als die Auflösung der Grafikkarte sein darf. Bewegt man die Maus zum Rand, wird der sichtbare Ausschnitt entsprechend angepaßt. Zu beachten ist, daß für jede Auflösung ein eigener virtueller Anzeigebereich definiert werden kann.

Bevor der virtuelle Bereich konfiguriert wird, muß man die entsprechende Frage mit einem »y« beantworten. Danach wird man nach den Farbtiefen gefragt, für die virtuelle Bildschirme konfiguriert werden können. Will man beispielsweise die virtuelle Auflösung für die 16-Bit-Modi ändern, muß man eine 2 eingeben. Anschließend kann man die gewünschte maximale Auflösung spezifizieren.

Man kann immer nur jeweils einen Modus konfigurieren, danach gelangt man wieder zur Abfrage zurück, ob man eine virtuelle Größe einrichten will. Beantwortet man diese Frage mit »n«, muß man in der nächsten angeben, ob die gewählten Modi in Ordnung sind.

Bevor die Konfiguration abgeschlossen wird, läßt sich noch festlegen, mit welcher Farbtiefe das System gestartet werden soll. Sind bei dieser Farbtiefe mehrere Auflösungen angegeben, wird immer die erste in der Liste genommen. Sie können aber im Betrieb mit den Tastenkombinationen [Strg]+[Alt]+[-] und [Strg]+[Alt]+[+] mit den Plus- und Minuszeichen vom numerischen Block umgeschaltet werden. Will man aber als Standard-Einstellung eine Farbtiefe von 16 Bit wählen, gibt man an dieser Stelle eine 4 ein.

Zum Abschluß der Einstellungen erfolgt eine Sicherheitsabfrage, ob die Datei »/etc/X11/XF86Config« geschrieben werden soll. Diese Frage kann man mit [n] beantworten, dann wird die Konfigurationsdatei »/usr/X11R6/lib/X11/XF86Config« verwendet[1]. Nach ihr wird man im nächsten Schritt gefragt, so daß man spätestens jetzt ein [y] eingeben sollte.

Damit ist die Konfiguration beendet und man kann versuchen, X Window zu starten. Was man dazu tun muß, ist im Kapitel zu den alternativen Konfigurationsmöglichkeiten beschrieben.

12.3.2 xf86cfg

Viel einfacher als mit dem textbasierten System geht die Konfiguration mit dem grafischen Programm. Das bei XFree 4 mitgelieferte Programm *xf86cfg* löst das unter Xfree 3 verwendete *XF86Setup* ab, das bei NetBSD aus den Packages in der Regel nachinstalliert werden mußte. *xf86cfg* hingegen wird standardmäßig mit installiert, so daß es sofort einsetzbar ist.

Voraussetzung dafür, daß die Konfiguration mit *xf86cfg* funktioniert, ist ein bislang nicht gestartetes X-Window-System, also nachdem NetBSD beispielsweise frisch installiert ist. Gibt man nach dem Einloggen an der Kommandozeile

```
xf86cfg
```

ein, wird ein Standard-X-Window-System mit der VGA-Auflösung 640 x 480 gestartet. Der Fenstermanager ist twm, der bei den X-Paketen standardmäßig mit installiert wird.

Nach dem Start befinden sich zwei Fenster auf dem Desktop, wobei mit dem linken das System konfiguriert wird. Das rechte stellt die Zugriffsmöglichkeit per Tastatur ein, dazu später mehr.

Das linke Fenster ist das Hauptfenster für die Konfiguration. Es stellt eine Reihe von Möglichkeiten bereit, hier soll ein Weg aufgezeigt werden, mit welchen Schritten die Konfiguration erfolgen kann.

---

[1] X prüft beim Starten des Servers zuerst, ob die Datei »/etc/X11/XF86Config« vorhanden ist. Fehlt sie, wird »/usr/X11R6/lib/X11/XF86Config « verwendet.

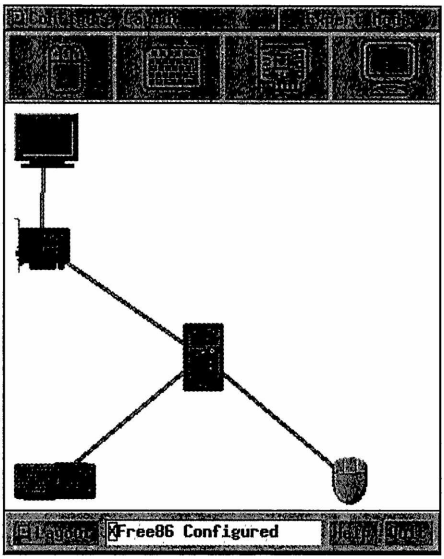


Abb. 12.1:
Das Hauptfenster der Konfiguration von X Window

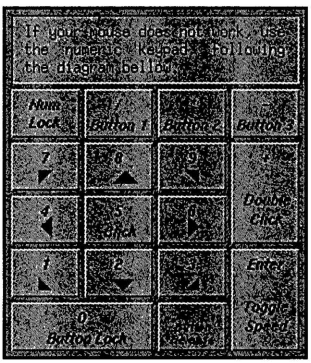


Abb. 12.2:
Die Bedienung mit der Tastatur

Die zuerst einstellbaren Optionen sieht man grafisch aufbereitet im Fenster. Dazu gehören die Tastatur, die Maus, die Grafikkarte und der Monitor, entsprechend der Buttons unterhalb des Menüs. Zur Konfiguration klickt man entweder der Reihe nach die einzelnen Buttons an und wählt aus dem angezeigten Menü den Punkt aus, der mit *Configure* beginnt, oder man geht über das lokale Menü der einzelnen Elemente im Fenster, das man über die rechten Maustaste erhält. Das setzt natürlich voraus, daß das System bereits die Maus korrekt erkannt und konfiguriert hat, was in der Regel der Fall sein sollte (es klapptauch problemlos mit einer USB-Maus). Funktioniert das nicht, kann man die Punkte mit Hilfe der Tastatur ansteuern. Wie das funktioniert, sieht man im rechten Fenster.

Die Bedienung erfolgt über den numerischen Tastenblock. Die Tasten in der Reihe neben [NumLock] entsprechen den Maustasten. Mit den Zifferntasten 1 bis 9 ohne die 5 wird der Cursor positioniert. Das Auszulösen von Aktionen, die einem Klick mit einer der Maustasten entsprechen, erfolgt über die Taste 5. Zuvor mußte die Maustaste gewählt werden. Analog zu Taste 5 sind auch Doppelklicks möglich; sie werden durch das Pluszeichen ausgelöst.

Da viele Operationen mit einer gedrückten Maustaste geschehen und auf der Tastatur keine zwei Tasten des Blocks gleichzeitig betätigt werden können, simuliert die 0 das Gedrückthalten der zuvor auszuwählenden Maustaste. Die Taste bleibt solange gedrückt, bis der Dezimalpunkt (beziehungsweise das Komma-Zeichen) auf der Tastatur eingegeben wird. Abschließend wird mit der Eingabetaste die Geschwindigkeit geändert, mit der der simulierte Mauszeiger bewegt wird.

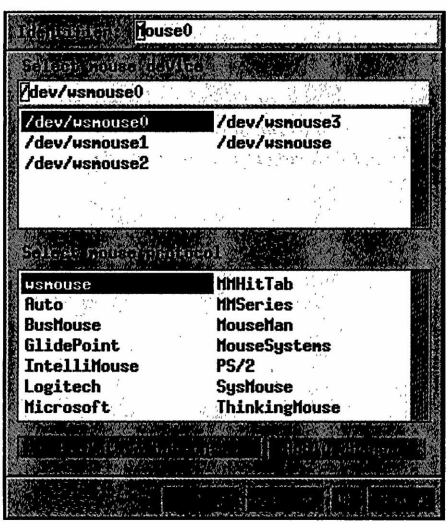


Abb. 12.3:
Die Konfiguration der Maus

Begonnen wird die Konfiguration von X Window mit der Einstellung der Maus. Funktioniert diese bereits, muß man die Optionen nur noch überprüfen. Ganz oben auf der Seite kann ein String zur Bezeichnung der Maus angegeben werden. Dieser ist aber nur dann wichtig, wenn mehrere Mäuse verwendet werden sollen, so daß man in der Regel die Angabe *Mouse0* übernehmen kann. Ansonsten wird auf dieser Seite das Device eingestellt, über das die Maus angesprochen werden soll, und man wählt das zur Kommunikation verwendete Protokoll. Ist man sich nicht sicher, stellt man die Option *Auto* ein, wodurch versucht wird, die entsprechende Wahl automatisch auszuführen.

Im diesem Dialog wird auch eingestellt, ob die dritte Maustaste emuliert werden soll. Das ist nur dann wichtig, wenn keine 3-Tasten-Maus angeschlossen ist. Zusätzlich testet man die gewählten Einstellungen über den Button *Apply Changes*. Sollte die Maus danach nicht mehr funktionieren, sollte man so lange andere Optionen wählen, bis die

passenden gefunden ist. Sind die Einstellungen abgeschlossen, kommt man mit *Ok* wieder in das Hauptfenster zurück. In ihm wird im nächsten Schritt die Konfiguration der Tastatur aufgerufen.

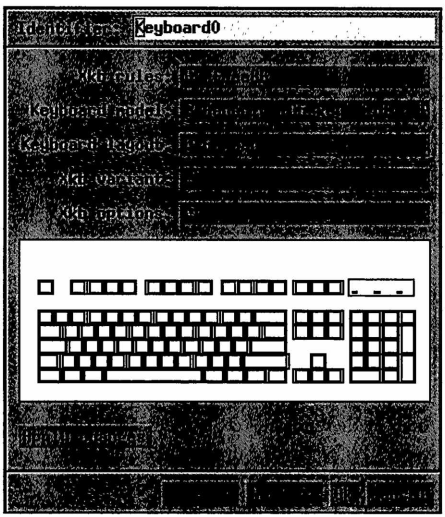


Abb. 12.4:
Die Konfiguration der Tastatur

Man sollte als erstes im oberen Feld einen Bezeichner für die Tastatur festlegen. Solange man nur eine hat, läßt man es einfach bei der Voreinstellung *Keyboard0*. Darunter werden die verwendeten Regeln eingestellt. Normalerweise dürfte hier die Option *xfree86* einzustellen sein, es sei denn, man betreibt NetBSD beispielsweise auf einer Sparc. In diesem Fall wäre die richtige Einstellung vermutlich *sun*. Danach wird das verwendete Tastaturmodell eingestellen, in dem festgelegt ist, wieviele Tasten die Tastatur besitzt. Für alle modernen Tastaturen ist die im vorigen Bild gezeigte Einstellung (*Generic 105-key (Intl) PC*) die richtige (zählen Sie die Tasten selbst...). Zusätzlich muß man darunter noch einstellen, welches Layout verwendet werden soll. Dafür gibt man die Sprache an, so daß als Möglichkeit nur noch *German* übrigbleibt.

Sollte man Probleme mit einigen Tasten haben, ist unter *Variant* noch eine Sonderbelegung möglich. Außerdem kann man bestimmten Tasten spezielle Funktionen zuweisen. Die entsprechenden Optionen befinden sich unter *Xkb options*. Für erste Tests muß man hier aber nichts tun, man kann nämlich bei Bedarf die Konfigurationsdatei noch mit einem Editor ändern. Über den den Button *Apply* werden die Einstellungen sofort wirksam und stehen für die weitere Konfiguration bereit.

Schließt man das Fenster wieder mit Ok, gelangt man wieder in das Hauptfenster zurück. Darin wird der nächste Punkt der Konfiguration in Angriff genommen, die Einstellung der Grafikkarte.

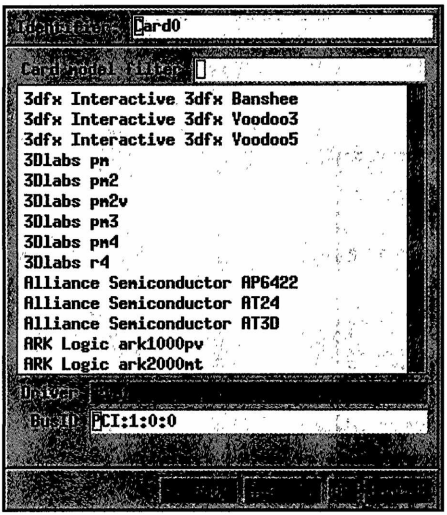


Abb. 12.5:
Die Konfiguration der Grafikkarte

Ganz oben auf der Seite befindet sich wieder ein Eingabefeld, in das eine Bezeichnung für die Grafikkarte eingegeben wird. Auch in diesem Fall reicht die Standardbezeichnung aus. Darunter befindet sehen sich eine Liste mit den unterstützen Grafikkarten, aus der die im jeweiligen Rechner ausgewählen wird. Die Anzahl der Karten in der angezeigten Liste wird eingeschränkt, indem im Feld *Card model filter* ein entsprechender String angegeben wird. Damit er wirksam wird, muß man noch die Eingabetaste drükken. Allerdings ist die Funktionsweise dieses Filters recht undurchsichtig, so daß man sich nicht darauf verlassen sollte. Ist die richtige Grafikkarte nicht enthalten, ist das noch problemlos, denn im Feld *Driver* ist manuell der passenden Treiber auszuwählen. Interessanterweise wurde hier bei der zum Test verwendeten ATI-Karte der Treiber *ati* angezeigt, der aber in der Liste, die nach Anklicken des entsprechenden Buttons erscheint, nicht vorhanden ist. Er funktioniert später (zumindest auf dem Rechner des Autors), so daß die Einstellung korrekt war, aber merkwürdigerweise fehlen auch die ATI-Karten in der Liste. Will man also eine ATI-Karte einsetzen, ist es ratsam, es an dieser Stelle bei der Voreinstellung zu belassen.

Das gilt generell auch für die Bus-ID, die im untersten Feld angezeigt wird. Sie wird vom System bestimmt und sollte möglichst nicht geändert werden. Über die ID wird die Karte eindeutig identifiziert und angesprochen. Sie ist besonders dann wichtig, wenn der Rechner über mehr als eine Grafikkarte verfügt.

Sind die Einstellungen abgeschlossen, wird die Seite mit *Ok* wieder verlassen. Danach wird der Monitor konfiguriert, wozu die nachfolgende Seite auf dem Bildschirm erscheint.

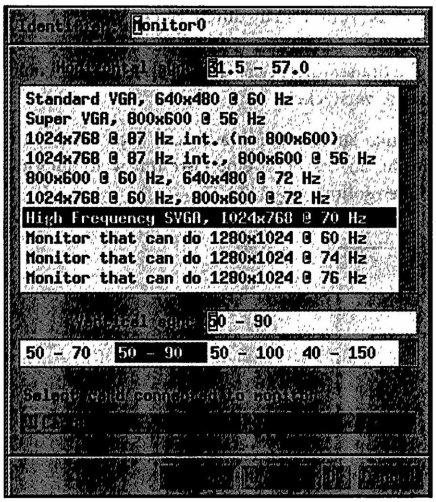


Abb. 12.6:
Die Konfiguration des Monitors

Bevor die weiteren Einstellungen ausgeführt werden, gibt man oben auf der Seite wieder eine Bezeichnung für den Monitor an, wobei man es natürlich wieder bei der Standardbezeichnung belassen kann, sofern man nicht eine spezielle benötigt. Wichtig auf dieser Seite sind die Werte *Horizontal sync* und *Vertical sync*. In diesen Feldern werden Bereiche eingetragen, die man auch den Listen entnehmen kann, die jeweils darunter angezeigt werden. Zu beachten ist, daß mit dieser Wahl eventuell der Monitor einen Defekt erleiden kann, wenn er die für ihn gültigen Werte überschreitet. Welche das sind, entnimmt man dem zugehörigen Handbuch.

Andererseits sollte man die beiden Frequenzen so hoch wie möglich wählen, damit der Desktop mit einer ergonomischen Wiederholfrequenz betrieben werden kann. Ist man sich nicht sicher, hat sich die Wahl von *High Frequency SVGA, 1024x768 @ 70 Hz* bewährt. Damit wird ein Desktop von 1024x768 Bildpunkten mit ungefähr 70 Hertz betrieben, was heutzutage die meisten Monitore klaglos mitmachen sollten. Allerdings ist diese Bildwiederholfrequenz nicht gerade ergonomisch, aber für erste Tests reicht es aus.

Damit die Zuordnung des Monitors zur Grafikkarte reibungslos klappt, wird unten auf der Seite noch angegeben, welche Karte verwendet werden soll. Hier werden die Bezeichner der im System vorhandenen Grafikkarten angezeigt, wenn man den zugehörigen Button anklickt. Das ist der String, der im oberen Teil der Seite konfiguriert werden kann.

Sind alle Einstellungen ausgeführt, kommt man mit *Ok* wieder in das Hauptfenster zurück. Klickt man darin den Button *Quit* an, wird die Konfigurationsdatei geschrieben, worauf man durch eine entsprechende Abfrage aufmerksam gemacht wird. Doch leider fehlen in der Konfigurationsdatei einige wichtige Abschnitte, so daß der Versuch, mit dieser ein X-Window-System zu starten, nicht zum gewünschten Erfolg führt.

Die entsprechenden Konfigurationsschritte bestimmt man durch einen Vergleich der grafischen mit der Textversion. Dazu gehören beispielsweise die Festlegung der Auflösungen in den einzelnen Farbtiefen und die Standardfarbtiefe. Bleibt also nur die Frage zu klären, wo sich die entsprechenden Punkte verstecken.

Man erreicht sie über den Button oben links auf der Seite, auf dem nach dem Start *Configure Layout* zu finden ist. Klickt man ihn an, erscheint eine Liste mit weiteren Optionen, aus der man *Configure Screen* auswählen sollte.

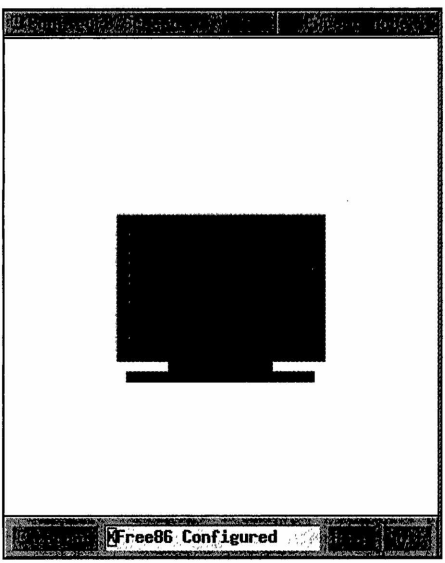


Abb. 12.7:
Die weitere Konfiguration des Monitors

Dadurch erscheint das vorige Bild (Abb. 12.7) auf dem Desktop, was nicht gerade viele Möglichkeiten zur Konfiguration bietet. Die zugehörigen Optionen erreicht man über das lokale Menü und Auswahl des Punkts *configure*. Dadurch ändert sich der Inhalt des Fensters wie in Abbildung 12.8.

Auf dieser Seite stellt man die zu verwendenden Auflösungen für den gewählten Bildschirm, dessen Bezeichnung man im oberen Eingabefeld sieht, ein. Dabei sind in der linken Liste die möglichen und in der rechten die gesetzten Auflösungen zu sehen, standardmäßig 640 x 480 Bildpunkte. Mit Hilfe der Rechts-/Links-Pfeile fügt man weitere Auflösungen hinzu beziehungsweise entfernt bereits gesetzte wieder. Da der Desktop standardmäßig mit der Auflösung gestartet wird, die an erster Stelle steht, kann man mit den anderen beiden Pfeilen die Reihenfolge der Auflösungen vertauschen.

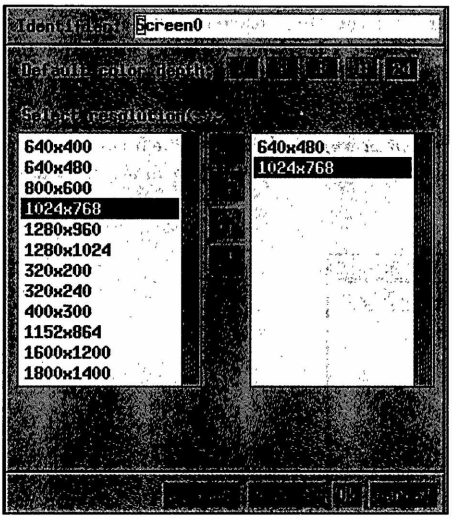


Abb. 12.8:
Die Konfiguration der Auflösungen

Oben auf der Seite besteht die Möglichkeit, die Farbtiefe einzustellen, mit der der Server gestartet werden soll. Zu beachten ist dabei, daß der auf der Karte verwendete Speicher auch einen Einfluß auf die maximale Farbtiefe und die höchstmögliche Auflösung hat. Die entsprechenden Zuordnungen sind Tabelle 12.3 zu entnehmen. Die bei 24 Bit Farbtiefe angegebenen Werte in Klammern würden einer 32-bittigen Farbtiefe entsprechen. Sie werden spezifiziert, da einige Karten auch bei 24 Bit Farbtiefe intern den Speicher mit 32 Bit verwalten, um die Zugriffe darauf schneller abwickeln zu können.

Auflösung	8-Bit-Farbtiefe	16-Bit-Farbtiefe	24-Bit-Farbtiefe
640x480	300 Kbyte	600 Kbyte	900 Kbyte (bzw.1200 Kbyte)
800x600	468,75 Kbyte	937,5 Kbyte	1406,25 Kbyte (bzw. 1875 Kbyte)
1024x768	768 Kbyte	1536 Kbyte	2304 Kbyte (bzw. 3072 Kbyte)
1152x864	972 Kbyte	1944 Kbyte	2916 Kbyte (bzw. 3888 Kbyte)
1280x1024	1280 Kbyte	2560 Kbyte	3840 Kbyte (bzw. 5120 Kbyte)
1600x1200	1875 Kbyte	3750 Kbyte	5625 Kbyte (bzw. 7500 Kbyte)

Tabelle 12.3: Die benötigten Speichergrößen

Auf der Konfigurationsseite kann man zwar alle Möglichkeiten einstellen, aber wenn der Speicher auf der Grafikkarte fehlt, verwendet der Server die nächstniedrige Auflösung und/oder Farbtiefe. Ist die Konfiguration abgeschlossen, kommt man mit *Ok* wieder auf die Seite zurück, auf der der Monitor dargestellt wird.

Dort wird der Button links oben (*Configure Screen*) angeklickt, wodurch ein Menü mit weiteren Punkten erscheint. Daraus wählt man den Punkt *Configure Modeline*, wodurch das nachfolgende Fenster angezeigt wird.

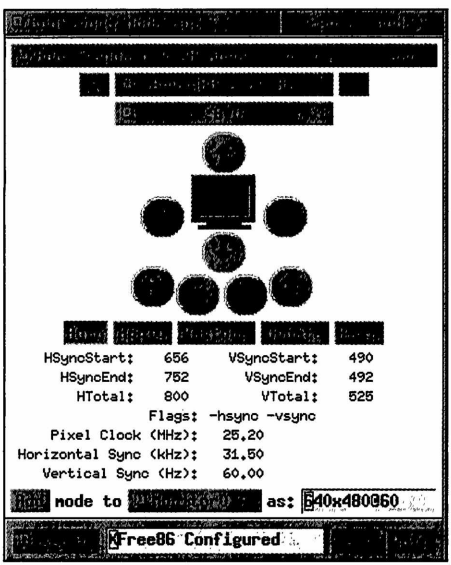


Abb. 12.9:
Die Konfiguration der Modeline

In diesem Schritt hat man die Möglichkeit, die genaue Justierung der einzelnen Auflösungen vorzunehmen. Das entspricht der Funktionalität des Programms *xvidtune*, das bei der Version 3 zu diesem Zweck zum Einsatz kam.

Im oberen Bereich wählt man die möglichen Auflösungen, wobei standardmäßig 640 x 480 bei 60 Hz aktiv ist. Wird ein entsprechender Button angeklickt, wird eine weitere angezeigt, die aber noch weniger Punkte darstellt. Leider fehlen die zuvor konfigurierten Auflösungen in der Liste, so daß man die Konfiguration an dieser Stelle gleich wieder verlassen kann, da vermutlich selten eine Auflösung von maximal 640 x 480 Bildpunkten eingesetzt wird. Das liegt daran, daß der Standardserver von *xf86cfg* die Nummer 8 besitzt, der normale X-Server aber unter der Nummer 0 läuft und während der Konfiguration natürlich nicht gestartet ist, weshalb man ihn an dieser Stelle auch nicht konfigurieren kann.

Damit ist im Prinzip die Konfiguration der Grafikkarte abgeschlossen. Klickt man den Button *Quit* an, wird man gefragt, ob man die Konfigurationsdatei »/usr/X11-R6/lib/X11/XF86Config« schreiben will. Man kann sie auf jeden Fall mit Ok beantwor-

ten, da nach diesen Konfigurationsschritten eine korrekte Konfigurationsdatei geschrieben wird. Auch die folgende Frage ist zu bejahen, so daß danach die Einstellung des X-Servers abgeschlossen ist. Alternativ – wenn Ihnen die Speicherung im Verzeichnis »/etc/X11« lieber ist, geben sie diesen Pfad und den Namen der Datei an.

Alternativ geschieht die Konfiguration des X-Servers auch über den Experten-Modus. Auf diesen kommt man im Hauptfenster über den Button oben rechts *Expert mode*. Klickt man ihn an, erscheint ein Fenster auf dem Desktop.

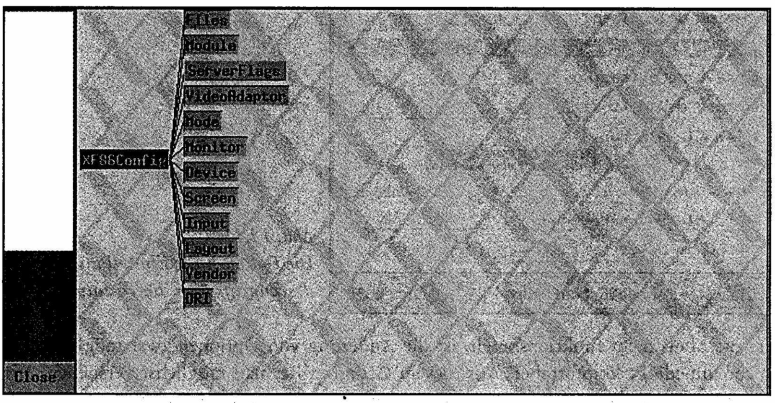


Abb. 12.10: Der Experten-Modus

Dort werden alle Konfigurationsmöglichkeiten in einer Baumstruktur grafisch dargestellt. Klickt man einen der Punkte an, expandiert dieser und zeigt weitere Optionen und Eingabefelder an. Werden diese Optionen angeklickt, werden weitere Punkte sichtbar. Macht man das bei mehreren, wird das ganze sehr unübersichtlich. Leider befinden sich auch einige Optionen, die für eine Konfiguration mitunter wichtig wären, jenseits der Desktop-Grenzen, so daß sie nicht einsetzbar sind. Da man keine höhere Auflösung einstellen kann und der Bildbereich nicht verschoben wird, wenn der Mauszeiger an den Rand kommt, ist der Expertenmodus zumindest für die Grundkonfiguration nicht empfehlenswert. Verlassen wird dieser Modus durch Auswahl von *Close*.

Damit sind fast alle Möglichkeiten angesprochen, aber eine besteht doch noch. Klickt man den linken oberen Button an und wählt den Punkt *Configure AccessX* aus, erscheint das in Abbildung 12.11 angezeigte Fenster auf dem Bildschirm.

Auf dieser Seite kann man beispielsweise einen Timeout setzen, nach dem eine Aktion automatisch aufgehoben wird. Außerdem ist es möglich, die Tastaturunterstützung zu aktivieren, obwohl auch die Maus korrekt funktioniert (*Enable AccessX*). Zusätzlich ist die maximale Geschwindigkeit des simulierten Mauszeigers und die Wiederholrate einstellbar.

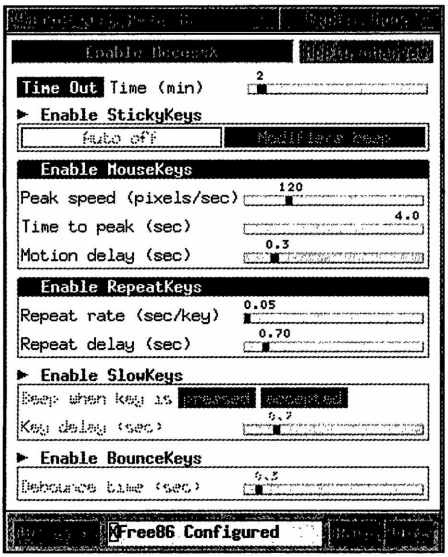


Abb. 12.11:
Die Optionen zur Einstellung des Zugriffs über die Tastatur

Ist die Maus nicht funktionsbereit, ist die Änderung von Optionen zwar möglich, aber nicht unbedingt komfortabel. Aus diesem Grund sollte man möglichst versuchen, erst die Maus über die Tastatur zu konfigurieren und die neuen Einstellungen mit *Apply changes* zu übernehmen. Danach läßt sich der weitere Konfigurationsvorgang bequem mit der Maus weiter ausführen.

12.3.3 Manuelle Bearbeitung

Man kann aber auch die Beispielkonfigurationsdatei »XF86Config.eg« aus dem Verzeichnis »/usr/X11R6/lib/X11« manuell anpassen. Dazu wird sie in *XF86Config* umbenannt und in einen Editor geladen. Anschließend kann man sie bearbeiten.

Wie auch unter XFree86 Version 3 ist die Konfigurationsdatei in Sektionen aufgeteilt, die mit dem Schlüsselwort *Section* beginnen und mit *EndSection* aufhören. Zu beachten ist, daß die Sektionsnamen in Anführungszeichen eingeschlossen werden müssen, denn sonst werden sie nicht richtig interpretiert. Ansonsten werden Kommentare mit einem Doppelkreuz eingeleitet, wie es bei Unix-Systemen üblich ist. Die Reihenfolge der Sektionen ist bei XFree86 Version 4 nicht mehr relevant, so daß sie im Prinzip in beliebiger Reihenfolge in der Datei stehen können.

Beim Betrachten des Inhalts der Beispieldatei sieht man, daß zuerst die Pfade auf benötigte Dateien gesetzt werden. Da es anders als bei Linux und seinen vielen Distributionen nur eine Version von NetBSD gibt, kann man sich diese Angabe im Prinzip sparen, da dann die Standardeinstellungen zum Zug kommen, die auf jeden Fall passen. Etwas anders sieht es aus, wenn man zusätzliche Schriften in Verzeichnissen installiert hat, die nicht dem Standard entsprechen. Dann ist es ratsam, die Pfade auf die Schriften explizit anzugeben, wie hier geschehen.

Danach folgt eine optionale Sektion mit dem Namen *Module*. In ihr können weitere Erweiterungsmodule geladen werden, die zum Betrieb des Servers benötigt werden. Das Modul *dbe* beispielsweise stellt eine Doppelpuffer-Funktion für den Bildschirm bereit. Zwischen diesen beiden Puffern kann sehr schnell umgeschaltet werden, so daß damit diverse Effekte möglich sind. Außerdem gehören diejenigen dazu, die für die Anzeige von Typ1- und FreeType-Schriften eingesetzt werden.

In der nun folgenden Sektion *ServerFlags* befinden sich verschiedene Einstellungen, die für den kompletten Server gelten (sie sind in der Regel alle auskommentiert). Dazu gehören beispielsweise Optionen, um bestimmte Tastenkombinationen zu deaktivieren, die normalerweise spezielle Bedeutungen für den Server haben, etwa [Strg]+[Alt]+[Backspace] für dessen Abbruch. Weiterhin gibt es hier Optionen, mit denen das Abschaltverhalten des Monitors beeinflußt wird.

Anschließend werden die Eingabegeräte näher spezifiziert. Dazu gehören sowohl die Tastatur, die mit dem Treiber (Driver) *Keyboard* angesprochen wird, als auch die Maus. Beide werden über Sektionen mit dem Namen *InputDevice* definiert. Zusätzlich muß man der Tastatur einen Bezeichner (Identifier) verpassen, der sie eindeutig identifiziert. Damit die Tastatur vernünftig funktioniert, müssen noch diverse Einstellungen vorgenommen werden. Dazu gehören beispielsweise die Angabe der Anzahl und der Anordnung der Tasten.

Nachdem die Tastatur eingestellt ist, geschieht das auch für die Maus. Es beginnt mit dem Setzen eines Bezeichners (Identifier) für die Maus gefolgt von der Spezifikation des Treibers (Driver). Weiterhin ist die Angabe des Protokolltyps erforderlich, mit dem versucht wird, mit der Maus eine Verbindung aufzubauen. Zu beachten ist, daß über die Wahl *auto* der Server gezwungen wird, den Typ automatisch herauszufinden. Das nimmt etwas Arbeit ab und funktioniert meist ohne Probleme. Außerdem muß man noch angeben, über welche Schnittstelle die Maus angesprochen werden soll. In diesem Fall ist es die Gerätedatei »/etc/mouse«, die normalerweise ein Link auf die passende Gerätedatei ist. Da bei einem X-Window-System die dritte Maustaste (die mittlere) in der Regel mit einer Funktion belegt ist, ist zum Abschluß der Maus-Sektion die Angabe (*Emulate3Buttons*) zu finden, daß die dritte Maustaste zu simulieren ist. Im späteren Betrieb geschieht das über das gleichzeitige Drücken beider Maustasten. Man sieht dann beispielsweise das zugehörige Menü auf dem Bildschirm. Will man diese Funktion nutzen, muß man das Doppelkreuz am Anfang der Zeile entfernen.

In der Beispieldatei wird eine weitere Maus konfiguriert. Der Server kann sie anhand der Bezeichner (Identifier) voneinander unterscheiden. Das gilt auch, wenn man mehrere Tastaturen konfigurieren will, sofern man jede anders benennt. Macht man das nicht, führt das zu einer entsprechenden Fehlermeldung.

Daran schließt sich die Beschreibung des Monitors an. Auch für ihn braucht man einen Bezeichner, den man sich merken muß. Danach folgen noch die Angaben der Bereiche für die horizontalen und vertikalen Synchronisationsbereiche, die man der Monitor-Dokumentation entnimmt. Beachten Sie, daß falsche Angaben unter Umständen zur Beschädigung des Monitors führen können.

Zusätzlich kann man an dieser Stelle spezielle Zeilen angeben, die genau festlegen, wie der Monitor angesteuert werden soll. Sie beginnen mit dem Wort »Mode« und legen

die Werte für das Timing einer Auflösung fest. Will man mehrere Auflösungen, muß man mehrere solche Zeilen angeben.

Welcher Treiber die Grafikkarte ansteuern soll, gibt man in der Sektion *Device* an. Wie alle anderen Geräte benötigt dieser ebenfalls einen Bezeichner (Identifier). Mit dem Befehl *Driver* wird dann der eigentliche Treiber festgelegt, also beispielsweise *ati*. Danach kann man noch spezielle Eigenschaften festlegen, die für den Treiber relevant sind, wie beispielsweise die Größe des Bildspeichers.

Die Beschreibung der Bildschirme und der verwendeten Auflösungen schließt sich an. Das Wort Bildschirm (*Screen*) beschreibt hier nicht einen physikalischen Gegenstand, sondern die Darstellung auf dem Monitor. Dazu muß man dem Bildschirm zuerst wieder einen Bezeichner verpassen, der später noch gebraucht wird. Als nächstes ist anzugeben, welchen Treiber (Device) und welchen Monitortyp (Monitor) man ansprechen will. Dazu benötigt man die Bezeichner (und nicht die Namen der Treiber selbst) der jeweiligen Geräte, die man zuvor vergeben hat. Außerdem muß man noch angeben, mit welcher Standardfarbtiefe (*DefaultDepth*) die Oberfläche angezeigt werden soll, also beispielsweise 16 Bit. Damit können dann also 65536 Farben dargestellt werden.

Eine genauere Spezifikation der Auflösung wird im folgenden vorgenommen. Das geschieht in einer Untersektion (*SubSection*), die mit dem Schlüsselwort *EndSubsection* beendet wird. In dieser wird eine Farbtiefe (*Depth*) definiert, woraufhin man die Modi beziehungsweise Auflösungen angibt, die dargestellt werden sollen. Dabei ist es auch möglich, mehrere Modi nacheinander zu spezifizieren. Wenn der Treiber es zuläßt, kann man zwischen diesen mit den Tastenkombinationen [Alt]+[Strg]+[+] und [Alt]+[Strg]+[-] umschalten. Zu beachten ist, daß die Plus- und Minuszeichen nur die auf dem numerischen Tastenblock sind! Auch wenn man mehrere verschiedene Auflösungen angibt, bedeutet das leider noch lange nicht, daß tatsächlich alle verfügbar sind. Mit welcher Auflösung man dann startet, hängt stark von der verwendeten Hardware ab.

Abschließend wird noch die linke obere Ecke des ursprünglichen Bildschirms (*ViewPort*) gesetzt, allerdings ist sie nur dann sinnvoll, wenn die virtuelle Bildschirmauflösung nicht mit der ursprünglichen Startauflösung übereinstimmt.

Damit die ganzen, zuvor gemachten Einstellungen funktionieren können, muß man zum Schluß der Konfigurationsdatei noch eine Sektion angeben, die als *ServerLayout* bezeichnet wird. Dort muß erstes wieder einen Bezeichner (Identifier) gesetzt werden. Anschließend gibt man den zu verwendenden Bildschirm (*Screen*) und die einzusetzenden Eingabegeräte (*InputDevice*) an. In allen Fällen sind dafür wieder die gesetzten Namen zu verwenden, nicht die Namen der Treiber selbst.

Da man im Layout mehrere Mäuse und Tastaturen angeben kann, ist noch festzulegen, was die primären Eingabegeräte sind. Dazu dienen die beiden Angaben *CorePointer* und *CoreKeyboard*. Fehlen sie, weiß der Server nicht, welche Geräte er verwenden soll. Man sieht in einem solchen Fall eine Fehlermeldung und der Server startet nicht.

Gibt es mehrere ServerLayouts in der Datei, wird vom Server dasjenige verwendet, das er als erstes in der Datei findet. Man kann damit durch die Reihenfolge festlegen, mit welchem Layout der Server starten soll.

Ist alles passend eingestellt, startet das X-Window-System durch Eingabe von

```
# startx
```

an der Kommandozeile. Auf dem Monitor sollte ein Bild zu sehen sein. Ist das nicht der Fall, sollte man in der Konfigurationsdatei kontrollieren, ob dort alles korrekt eingestellt ist.

Sieht man Fehlermeldungen beim Aufruf und startet der Server gar nicht, müssen diese natürlich zuerst beseitigt werden. In der Regel wird relativ genau mitgeteilt, an welcher Stelle in der Datei man suchen muß, so daß es bei der Beseitigung eigentlich wenig Probleme geben sollte.

Wenn mehrere Auflösungen aktiviert sind, wird immer diejenige angewendet, die in der Liste an erster Stelle kommt (mitunter wird auch die mit der höchsten Anzahl an Bildpunkten eingesetzt). Umzuschalten ist die Auflösung mit den Tastenkombinationen [Alt]+[Strg]+[+] beziehungsweise [Alt]+[Strg]+[-] (die Plus- und Minustasten des numerischen Blocks), so daß man zumindest immer auf eine zurückschalten kann (VGA), die ein Bild liefert, oder man bricht den Server mit [Alt]+[Strg]+[Backspace] ab. Man kann dann so lange probieren, bis die gewünschte Auflösung gefunden ist.

12.3.4 Beispiel

Hier ein Beispiel einer automatisch vom Programm *xf86cfg* generierten Konfigurationsdatei *XF86Config*. Sie könnte wie folgt aussehen. Zu beachten ist dabei, daß die Reihenfolge der einzelnen Sektionen nicht festgelegt ist, sie also an beliebiger Stelle auftauchen können.

```
Section "ServerLayout"
    Identifier     "XFree86 Configured"
    Screen      0  "Screen0" 0 0
    InputDevice    "Mouse0" "CorePointer"
    InputDevice    "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
    RgbPath      "/usr/X11R6/lib/X11/rgb"
    ModulePath   "/usr/X11R6/lib/modules"
    FontPath     "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath     "/usr/X11R6/lib/X11/fonts/Speedo/"
    FontPath     "/usr/X11R6/lib/X11/fonts/Type1/"
    FontPath     "/usr/X11R6/lib/X11/fonts/CID/"
    FontPath     "/usr/X11R6/lib/X11/fonts/75dpi/"
    FontPath     "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection
```

```
Section "Module"
    Load  "dbe"
    Load  "extmod"
    Load  "glx"
    Load  "xtrap"
    Load  "speedo"
    Load  "type1"
EndSection

Section "InputDevice"
    Identifier  "Keyboard0"
    Driver      "keyboard"
    Option      "XkbRules"  "xfree86"
    Option      "XkbModel"  "pc105"
    Option      "XkbLayout" "de"
EndSection

Section "InputDevice"
    Identifier  "Mouse0"
    Driver      "mouse"
    Option      "Protocol" "wsmouse"
    Option      "Device" "/dev/wsmouse0"
EndSection

Section "Monitor"
    Identifier   "Monitor0"
    VendorName   "Monitor Vendor"
    ModelName    "Monitor Model"
    HorizSync    31.5 - 57.0
    VertRefresh  50.0 - 90.0
EndSection

Section "Device"
        ### Available Driver options are:-
        ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
        ### <string>: "String", <freq>: "<f> Hz/kHz/MHz"
        ### [arg]: arg optional
        #Option     "accel"                 # [<bool>]
        #Option     "crt_screen"            # [<bool>]
        #Option     "composite_sync"        # [<bool>]
        #Option     "hw_cursor"             # [<bool>]
        #Option     "linear"                # [<bool>]
        #Option     "mmio_cache"            # [<bool>]
        #Option     "probe_clocks"          # [<bool>]
        #Option     "reference_clock"       # <freq>
        #Option     "shadow_fb"             # [<bool>]
        #Option     "sw_cursor"             # [<bool>]
```

```
    Identifier  "Card0"
    Driver      "ati"
    VendorName  "ATI"
    BoardName   "Mach64 GZ"
    ChipSet     "ati"
    ChipId      0x475a
    ChipRev     0x3a
    BusID       "PCI:1:0:0"
EndSection

Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
    Monitor    "Monitor0"
    DefaultDepth     16
    SubSection "Display"
        Depth     1
    EndSubSection
    SubSection "Display"
        Depth     4
    EndSubSection
    SubSection "Display"
        Depth     8
    EndSubSection
    SubSection "Display"
        Depth     15
    EndSubSection
    SubSection "Display"
        Depth     16
        Modes    "1024x768" "640x480"
    EndSubSection
    SubSection "Display"
        Depth     24
    EndSubSection
EndSection
```

Die Datei beginnt mit einer *ServerLayout*-Sektion. In ihr wird zusammengefaßt, welche weiteren Bestandteile der Datei zum Starten des Servers verwendet werden. Dazu gehören der Bildschirm, die Maus und die Tastatur. Die nächste Sektion *Files* legt die Pfade auf die zum Betrieb benötigten Dateien und Schriften fest.

Anschließend werden in der Sektion diverse Module geladen. Dies sind *dbe, extmod, glx, record, xtrap, speedo* und *type1*. Die Bedeutung der Module ist der nachfolgenden Tabelle zu entnehmen.

Modul	Beschreibung
dbe	Doppelpuffer für den Bildschirm.
extmod	Verschiedene Server-Erweiterungen.
glx	Hardwarebeschleunigte 3D-Darstellung.
speedo	Aufbereitung von Schriften des Typs Speedo.
type1	Aufbereitung von Typ1-Schriften.
xtrap	DEC-XTRAP-Erweiterung.

Tabelle 12.4: Die Beschreibung der geladenen Module

Danach werden die Eingabegeräte beschrieben, wobei die erste Angabe die Tastatur betrifft. Hier werden der Treiber spezifiziert und anschließend befinden sich die Angaben für die zu verwendende Regeln, das Modell und das Layout.

Es schließt sich die Beschreibung der Maus an. In der entsprechenden Sektion werden der Treiber, das Protokoll und das Device spezifiziert. In diesem Fall wird der abstrakte Treiber *wsmouse* verwendet, der in Verbindung mit dem *wscons*-Framework steht. Dieser ist mit spezifischen Hardware-Treibern verbunden.

In der Beispieldatei folgt danach die Beschreibung des Monitors. In der zugehörigen Sektion *Monitor* werden einige Strings zur Beschreibung festgelegt. Außerdem finden sich die Angaben für die Bereiche der horizontalen und vertikalen Bildwiederholfrequenzen.

Die Sektion *Device*, die nun kommt, legt diverse Einstellungen für die Grafikkarte fest. Dazu gehören beispielsweise die frei wählbare Bezeichnung der Karte, der zu verwendende Treiber und die Bus-ID. Außerdem findet man dort Angaben zur ID und zur Version des Chips, die man normalerweise vermutlich nicht eintragen würde. Da diese Datei aber automatisch generiert wurde, wurden diese Informationen automatisch ausgelesen.

In der Sektion *Screen* wird nun die Verbindung zwischen der Karte und dem Monitor gezogen, wobei dazu die Bezeichner verwendet werden. Außerdem wird die Standardfarbtiefe festgelegt, mit der der Server gestartet werden soll, in diesem Fall sind es 16 Bit. Danach folgen Untersektionen, in denen die Zuordnung der Farbtiefe zu den verfügbaren Auflösungen hergestellt wird.

Nur wenn tatsächlich eine Modes-Zeile vorhanden ist, kann die entsprechende Auflösung verwendet werden.

12.4 Start des Default-Systems

Nach der Konfiguration des X-Window-Systems wird es durch Eingabe von

```
# startx
```

an einem Kommandozeilenprompt gestartet. Paßt die Konfigurationsdatei zur jeweiligen Hardware, man also die richtigen Optionen angegeben hat, wird in den Grafikmodus geschaltet und man sieht einige Fenster auf dem Bildschirm.

Mit der Tastenkombination [Alt]+[Strg]+[F1] bis [Alt]+[Strg]+[F4] schaltet man auf eine Textkonsole zurück. Damit stehen nach wie vor die ersten vier Konsolen zur Verfügung. Zurück auf die grafische Oberfläche gelangt man mit [Alt]+[Strg]+[F5].

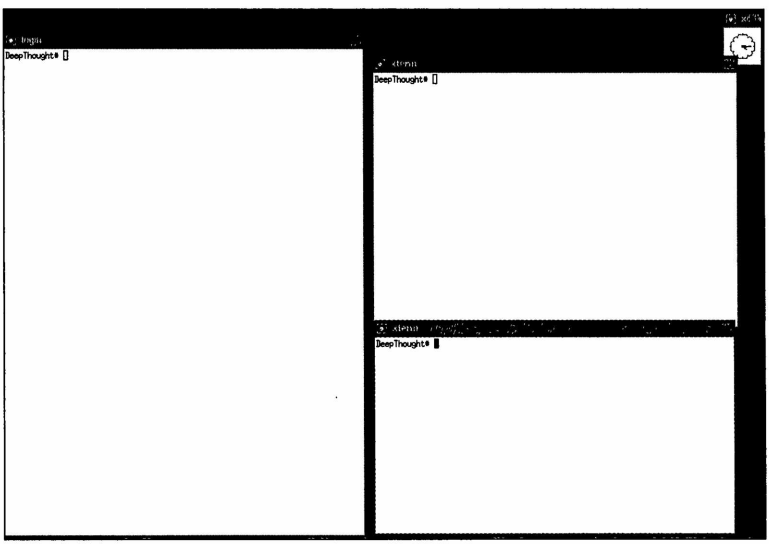


Abb. 12.12: Das Standardsystem nach der Installation

Leider wird standardmäßig nur der Fenstermanager *twm* installiert, der sehr wenig Komfort bietet. Aber man kann so zumindest erst einmal sehen, ob X Window überhaupt funktioniert. Auf langsamen Plattformen und Rechnern mit nur wenig Arbeitsspeicher, auf denen ein KDE- oder Gnome-System vielleicht gar nicht läuft, sollte man sich überhaupt überlegen, ob man den ansprichslosen twm nicht vielleicht doch einsetzen will.

Wenn man dann doch auf einen anderen Fenstermanager umsteigen will, muß man die entsprechenden Pakete nachinstallieren. In der Regel wird eine große Anzahl dieser Pakete mitgeliefert, so daß der Anwender die Qual der Wahl hat. Vermutlich werden Sie aber nur den Gnome-Desktop oder das KDE-System in die engere Auswahl einbeziehen, dennoch befindet sich in einem der nachfolgenden Kapitel zu jedem Fenstermanager eine Kurzbeschreibung, zu KDE und zu Gnome ist die Beschreibung ein wenig länger.

12.5 Die Konfigurationsdateien von X

Neben der schon erwähnten Konfigurationsdatei »/usr/X11R6/lib/X11/ XF86Config« gibt es noch eine Reihe weiterer Dateien, mit denen man das Verhalten des X-Window-Systems beeinflussen kann. Ein Großteil davon befindet sich im Verzeichnis »/usr/X11R6/lib/X11« und den jeweiligen Unterverzeichnissen, die die globalen Einstellungen bereitstellen. Welche das sind, steht in Tabelle 12.5 (außerdem gibt es noch einige Dateien im Heimatverzeichnis, deren Inhalt die globalen Einstellungen überschreiben).

Die für den Start eines X-Window-Systems wichtige Datei ist »/usr/X11R6/lib/X11/xinit/xinitrc«. Sie wird als letztes durch das Skript *startx* aufgerufen. In ihr werden Standardeinstellungen geladen wie die Ressourcen für den X-Server über das Programm *xrdb*. Außerdem erfolgt die Anpassung von Tastaturen und die Zuordnung der Maustasten. Dazu dient das Programm *xmodmap*. Abschließend werden der Fenstermanager *twm* und einige Terminals gestartet.

```
#!/bin/sh
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# start some nice programs

twm &
xclock -geometry 50x50-1+1 &
```

```
xterm -geometry 80x50+494+51 &
xterm -geometry 80x20+494-0 &
exec xterm -geometry 80x66+0+0 -name login
```

Dieses Verhalten kann man durch eine Datei mit dem Namen ».xinitrc« im Heimatverzeichnis verändern. In diesem Fall wird deren Inhalt ausgewertet und man beeinflußt so sehr unkompliziert, welcher Fenstermanager tatsächlich gestartet werden soll. Meist passiert dies durch ein einfaches

```
exec FENSTERMANAGER
```

so daß nur der zugehörige Fenstermanager gestartet wird, der den Rest dann selbst verwaltet, wie beispielsweise das Restaurieren des Desktops nach einem Neustart. Welche Programme beim Hochfahren gestartet werden, wird dann in den Konfigurationsdateien des jeweiligen Fenstermanagers gespeichert. Das sollte einem aber nicht davon abhalten, beispielsweise in der Konfigurationsdatei auch noch andere Programme mit anzugeben, wenn es denn nötig sein sollte. Der Eintrag muß nur vor dem Aufruf des Fenstermanagers erfolgen.

12.6 Nachträgliche Konfiguration

Auch wenn man nach der Konfiguration ein Bild auf dem Bildschirm sieht, kann es mitunter notwendig sein, diverse Feinarbeiten auszuführen. Meist ist es erforderlich, die Größe und die Lage des Bilds zu justieren, sofern man das nicht direkt am Monitor regelt.

12.6.1 xf86cfg

Das Programm, mit dem die Grundkonfiguration des X-Window-Systems ausgeführt wurde, läßt sich auch für die Feineinstellungen verwenden. Dazu startet man es an der Kommandozeile, wobei man aber die Lage der Konfigurationsdatei mit dem absoluten Pfad angeben muß.

```
# xf86cfg -xf86config /usr/X11R6/lib/X11/XF86Config
```

Vergißt man die Angabe der Option *-xf86config* oder gibt man nur einen relativen Pfad an, wird eine Fehlermeldung zurückgegeben, daß die Konfigurationsdatei nicht gefunden werden kann.

Verzeichnis	Beschreibung
/usr/X11R6/lib/X11/app-defaults	In diesem Verzeichnis befinden sich Dateien mit Beschreibungen über Aussehen und verwendete Farben. Außerdem sind hier die Konfigurationsdateien mit den Standardeinstellungen gespeichert.
/usr/X11R6/lib/X11/fs	Hier ist die Konfiguration des Font-Servers gespeichert.
/usr/X11R6/lib/X11/lbxproxy	Die Konfigurationsdateien des Proxies *lbxproxy* sind hier gespeichert.
/usr/X11R6/lib/X11/proxymgr	In diesem Verzeichnis befinden sich die Konfigurationsdatei(en) des Proxy-Managers.
/usr/X11R6/lib/X11/rstart	Hier sind die Konfigurationsdateien eines Remote Start Clients (*rstart*) zu finden.
/usr/X11R6/lib/X11/twm	Dieses Verzeichnis enthält die Konfigurationsdatei des Fenstermanagers *twm*.
/usr/X11R6/lib/X11/xdm	Die Konfigurationsdateien des X-Display-Managers *xdm* sind in diesem Verzeichnis enthalten. Sie werden teilweise auch vom *kdm* verwendet.
/usr/X11R6/lib/X11/xinit	Hier steht die Konfigurationsdatei »xinitrc« für die Initialisierung des X-Window-Systems.
/usr/X11R6/lib/X11/xkb	In diesem Verzeichnis sind Einstellungen zur Tastatur enthalten.
/usr/X11R6/lib/X11/xserver	Die Sicherheitseinstellungen für den Server sind in diesem Verzeichnis gespeichert.
/usr/X11R6/lib/X11/xsm	Das Verzeichnis enthält die Einstellungen des Sitzungs-Managers *xsm*.

Tabelle 12.5: Die globalen Konfigurationsdateien

Nach dem Start des Programms sieht man die bereits bekannten Fenster auf dem Bildschirm. Nur wird diesmal nicht auf eine VGA-Auflösung umgeschaltet und wenn man die einzelnen Optionen näher betrachtet, sieht man die aktuellen Einstellungen. Da jetzt eine höhere Auflösung verwendet wird, läßt sich auch der Expertenmodus halbwegs vernünftig nutzen.

Der Expertenmodus

Wenn man den Button *Expert Mode* anklickt, erscheint ein weiteres Fenster auf dem Bildschirm, in dem alle konfigurierbaren Bestandteile aufgezeigt werden. Man kann sich hier beispielsweise die Einstellungen für den Monitor oder die Grafikkarte anzeigen

lassen. Der Vorteil dabei besteht darin, daß man alles im Überblick hat, solange man immer nur eine Hauptoption zur Zeit bearbeitet. Will man dagegen zwei oder mehr bearbeiten, sollte man das nur nacheinander erledigen, denn sonst kann das Ganze ziemlich unübersichtlich werden.

Klickt man beispielsweise den Button *Monitor* an, werden die für diesen Punkt verfügbaren Optionen in Tabellenform angezeigt. Zusätzlich gibt es noch die beiden Optionen *Modeline* und *UseModes*, die durch Anklicken expandiert werden. Will man Werte ändern, kann dies direkt in der Tabelle geschehen.

Klickt man in der oberen Zeile der Tabelle den Button *remove* an, wird die komplette Tabelle mit allen Einträgen gelöscht. Dies passiert ohne Vorwarnung und es gibt auch keine *Undo*-Funktion, so daß man in diesem Modus sehr vorsichtig sein sollte.

Der andere Button *options* führt zu einem weiteren Dialog, in dem man zusätzliche Optionen für den Monitor angeben kann. Bestes Beispiel dafür ist die Option *DPMS*, durch die der Monitor nach geraumer Zeit abgeschaltet wird, wenn keine Eingaben mehr vorgenommen werden.

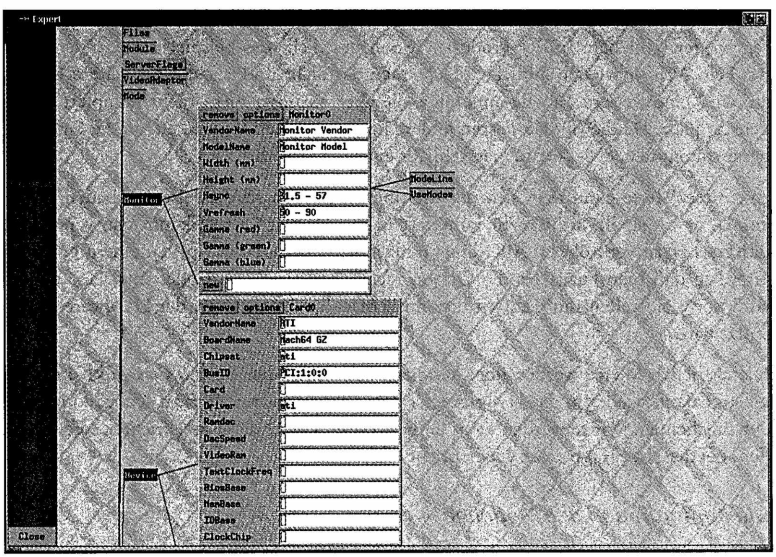


Abb. 12.13: Der Expertenmodus

In diesem Fenster kann man mit *add* weitere Optionen hinzufügen. Dazu wird eine entsprechende Liste angezeigt. Durch *remove* wird der gerade markierte Eintrag gelöscht. Der Button *update* aktualisiert die Liste der Optionen und mit *Help* hat man unter Umständen die Möglichkeit, sich Informationen zu der Option anzeigen zu lassen. Das Fenster selbst schließt man durch Auswahl des Buttons *Popdown dialog*.

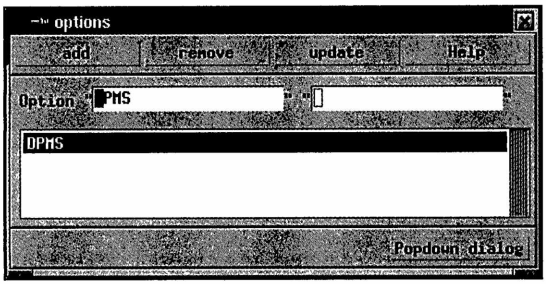


Abb. 12.14: Die Optionen für einen Monitor

Gibt man in der Zeile *new* unterhalb der Tabelle einen Namen ein und klickt man dann den Button an, erscheint eine neue Monitor-Tabelle, in der man eine andere Konfiguration ablegen kann. Diese kann später in der Sektion *ServerLayout* verwendet werden, indem man dort den gerade angelegten Bezeichner für den Monitor angibt. Auf diese Weise ist es sehr einfach möglich, auf einen anderen Monitor umzuschalten, sofern er vorher konfiguriert wurde.

Will man einen weiteren Eintrag in der Liste bearbeiten, sollte man den zuvor geöffneten wieder schließen. In der Monitor-Option geschieht das beispielsweise, indem man das Feld *Monitor* wieder anklickt. Macht man das nicht und klickt eine weitere Option an, wird eine weitere Tabelle angezeigt, die aber mitunter nicht mehr komplett in das Fenster paßt. Da die Bildlaufleisten fehlen, hat man keine Möglichkeit, an die Optionen heranzukommen, es sei denn, man schließt eine andere Tabelle.

Auf diese Weise kann man sich die einzelnen Optionen schrittweise anschauen. Sollte einmal eine Tabelle nicht mehr komplett in das Fenster passen, kann man versuchen, das Fenster zu vergrößern. Allerdings funktioniert das auch nur bis zu einem gewissen Grad, so daß man entweder die Option nicht bearbeiten kann oder man eine andere Auflösung einstellen muß, sofern dies noch machbar ist und von der Hardware unterstützt wird.

Die aktuellen Werte werden übernommen, wenn man den Button *Close* anklickt. Man gelangt dann wieder in das Hauptfenster von *xf86cfg*, das mit *Quit* zu schließen ist. Die entsprechenden Werte werden dann in die Konfigurationsdatei geschrieben, sofern man dem bei der Nachfrage nicht widerspricht.

Die Konfiguration der Modeline

Neben den üblichen Optionen, die auch schon bei der Grundkonfiguration eingestellt wurden und die jetzt mit den aktuellen Werten aus der Konfigurationsdatei gefüllt werden, kann man auch die Modeline konfigurieren, also die Ausrichtung des Bilds genauer einstellen. Dazu ist links oben der Button anzuklicken und die Option *Configure Modeline* auszuwählen. Auf dieser Seite wird im oberen Bereich der aktuelle Modus angezeigt. Mit den »<<«- und »>>«-Zeichen rechts daneben wird die Auflösung auf die nächste beziehungsweise vorige konfigurierte umgeschaltet. Darunter sieht man den derzeit verwendeten X-Server.

Die Pfeile rings um den Bildschirm beschreiben, was mit dem Bild passiert, wenn man den zugehörigen anklickt. Allerdings passiert das nur sofort, wenn der Button *Auto* gedrückt ist, also schwarz unterlegt erscheint. Ansonsten werden nur die Timing-Werte entsprechend verstellt, wobei die neuen Einstellungen dann wirksam werden, wenn man *Apply* oder *Test* anklickt.

Mit Hilfe der runden Buttons oberhalb von *Apply* läßt sich das Bild verbreitern, verschmälern oder in der Höhe verkleinern beziehungsweise vergrößern. Wie auch zuvor ändert sich nur dann etwas sofort, wenn die automatische Aktualisierung eingeschaltet ist. Ansonsten kann man mit *Test* prüfen, wobei nach geraumer Zeit wieder auf die vorige Einstellung zurückgeschaltet wird, oder man übernimmt sie mit *Apply* sofort.

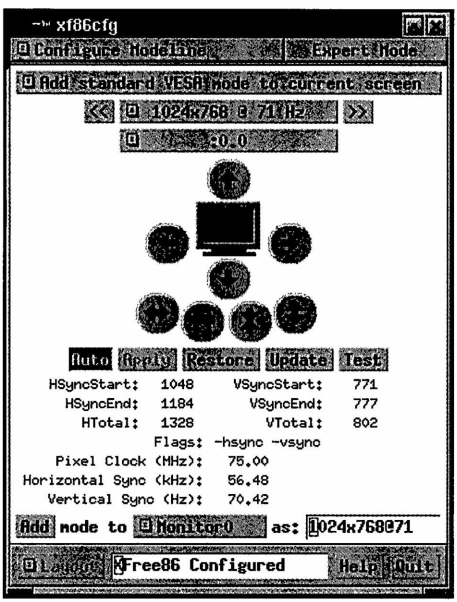


Abb. 12.15: Die Konfiguration der Modeline

Durch *Restore* wird der beim Start gültige Modus wiederhergestellt, was aber nicht unbedingt immer klappt, *Update* aktualisiert die Anzeigen der Daten im unteren Bereich der Seite.

Wenn man mit den Button experimentiert, kann es unter Umständen passieren, daß man an die Leistungsgrenzen des Monitors kommt. In einem solchen Fall erscheint plötzlich kein Bild mehr und man kann folglich auch nicht vernünftig einen anderen Button auswählen, um den Modus zurückzuschalten. Das wäre aber wichtig, da sonst der Monitor Schaden nehmen könnte. Also bleibt nichts anderes übrig, als die [Esc]-Taste zu betätigen, um in einen Modus zu schalten, in dem man wieder ein Bild sehen sollte. Anschließend kann man die Einstellung fortsetzen.

Sind sie abgeschlossen, muß man den Modus explizit in die Beschreibung des Monitors einfügen. Dazu ist es erforderlich, in der Mitte die Bezeichnung des Monitors zu wählen und anschließend den Button *Add* anzuklicken. Erst dann wird die entsprechende Zeile auch tatsächlich in die Konfigurationsdatei geschrieben, was passiert, wenn man zum Abschluß den Button *Quit* betätigt und die entsprechenden Nachfragen bejaht. Eine Monitor-Sektion könnte wie folgt aussehen, wenn die Modeline-Zeile konfiguriert wurde.

```
Section "Monitor"
Identifier     "Monitor[0]"
  VendorName   "AutoDetected"
  ModelName    "AutoDetected"
  HorizSync    30.0 - 82.0
  VertRefresh  50.0 - 70.0
  ModeLine "1024x768@70" 74.5 1024 1032 1208 1328 768 772 780 794 +hsync \
                                                                  +vsync
  Option       "DPMS"
EndSection
```

In diesem Fall ist zusätzlich noch die Option *DPMS* enthalten, mit der eine Abschaltung des Monitors erfolgt.

12.6.2 xvidtune

Auch wenn die Einstellungen mit *xf86cfg* vorgenommen wurden, kann man trotzdem das Programm *xvidtune* zum Feintuning hernehmen, das in Regel bei der Vorgängerversion XFFree86 3 eingesetzt wurde. Es wird gestartet, wenn man an einer Kommandozeile den Befehl

```
# xvidtune
```

eingibt. Nach dem Start wird ein Fenster angezeigt, in dem man die gewünschten Einstellungen ausführen kann. Gleichzeitig erscheint ein weiteres Fenster, das davor warnt, das Programm ohne ausreichendes Wissen zu einzusetzen. Andernfalls könnten der Monitor und/oder die Grafikkarte beschädigt werden. Die Warnung sollte man sehr ernst nehmen, besonders dann, wenn einem die Parameter, die man hier einstellen kann, unbekannt vorkommen.

Die eigentlichen Einstellungen werden erst dann vorgenommen, wenn das Fenster mit der Warnung durch Anklicken des Buttons *OK* geschlossen wurde. Wählt man dagegen den Button *Cancel* aus, werden beide Fenster geschlossen und man kommt wieder zurück an die Kommandozeile. Will man trotz der Warnung doch weitermachen, wird in der obersten Zeile angezeigt, welche Größe der Desktop momentan besitzt, links sieht man den horizontalen (*HDisplay*) und rechts den vertikalen Wert (*VDisplay*), also beispielsweise 1024 und 768 oder 640 und 480.

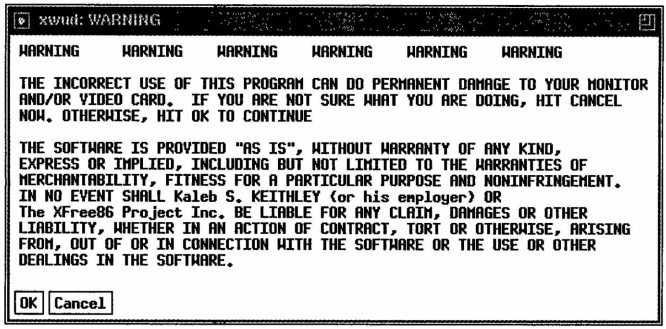


Abb. 12.16: Die Warnmeldung beim Start von xvidtune

Darunter befinden sich, jeweils getrennt für die horizontale und vertikale Synchronisation, diverse Schieberegler, mit denen einzustellen ist, an welcher Stelle einer Zeile die Synchronisationssignale beginnen und enden sollen und wie lang eine Zeile tatsächlich ist. Die restlichen Bereiche werden für Synchronisations- und andere videospezifische Zwecke benötigt, die aber natürlich nicht auf dem Bildschirm dargestellt werden (bei einem Fernsehbild ist übrigens in diesen Bereichen auch das Videotextsignal zu finden). Die Regler kann man jeweils mit der mittleren Maustaste (oder mit beiden gleichzeitig, sofern man nur eine Zweitastenmaus besitzt und die Emulation eingeschaltet haben) verschieben. Dazu bewegt man den Mauszeiger über einen der Regler. Drückt man jetzt die mittlere Maustaste und bewegt die Maus, sollte der Regler der Mausbewegung folgen. Gleichzeitig wird die Anzeige darüber mit der neuen Position aktualisiert. Läßt man die mittlere Maustaste los, behält der Schieberegler diese Position bei und die Einstellung ist abgeschlossen. Alternativ wählt man auch eine Position direkt an, indem man sie mit der mittleren Maustaste anklickt. Nach dem Loslassen sollte der Schieberegler an die gesetzte Stelle springen. Dieses Vorgehen ist besonders dann sinnvoll, wenn beispielsweise schnell zum Anfang oder zum Ende gesprungen werden soll.

In der nächsten Zeile gibt es jeweils vier Buttons, mit deren Hilfe die Position der Schieberegler verändert wird. In Tabelle 12.6 sind die Felder, die durch Anklicken eines Buttons verändert werden, dargestellt.

Insgesamt wird mit Hilfe der Buttons *Left*, *Right*, *Up* und *Down* die Lage des Bilds auf dem Desktop einfgestellt. Dieses geschieht in der Regel nicht sofort, sondern erst, wenn man den Button *Test* anklickt, doch dazu später mehr. Mit den anderen vier Buttons *Wider*, *Narrower*, *Shorter* und *Taller* stellt man die Größe des Bilds ein.

Darunter befindet sich in der linken Spalte ein Eingabefeld, in das spezielle Werte eingegeben werden können. Den hier angezeigten Wert sollte man in der Regel aber nicht verändern, es sei denn, das wird vom Kartenhersteller oder in einer entsprechenden Readme-Datei empfohlen.

Button	Beeinflußte Regler beziehungsweise Felder
Left	Erhöhung der Werte *HsyncStart* und *HSyncEnd*.
Right	Verminderung der Werte *HSyncStart* und *HSyncEnd*.
Wider	Verminderung des Werts *HTotal*, gleichzeitig Anpassung der Ausgaben in den Feldern *Horizontal Sync* und *Vertical Sync*.
Narrower	Erhöhung des Werts *HTotal*, gleichzeitig Anpassung der Ausgaben in den Feldern *Horizontal Sync* und *Vertical Sync*.
Up	Erhöhung der Werte *VsyncStart* und *VSyncEnd*.
Down	Verminderung der Werte *VSyncStart* und *VSyncEnd*.
Shorter	Erhöhung des Werts *VTotal*, gleichzeitig Anpassung der Ausgabe im Feld *Vertical Sync*.
Taller	Verminderung des Werts *VTotal*, gleichzeitig Anpassung der Ausgabe im Feld *Vertical Sync*.

Tabelle 12.6: Die Funktion der Steuerbutton im linken Fensterbereich von xvidtune

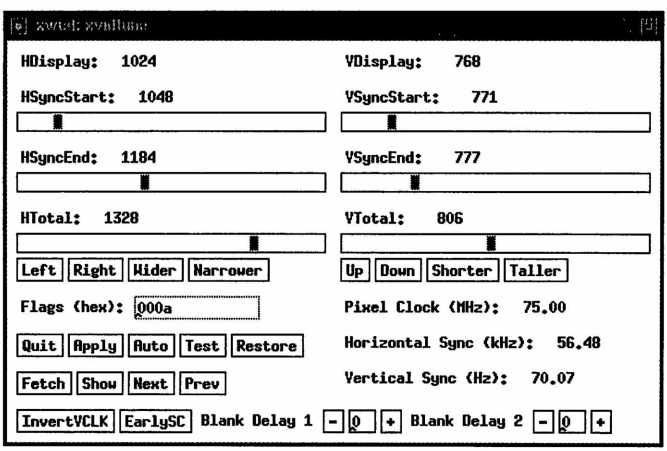


Abb. 12.17: Das Hauptfenster von xvidtune

In den beiden Zeilen unter diesem Feld sind zwei Reihen Buttons angebracht, deren Funktionsweise in der nachfolgenden Tabelle erläutert wird. Die Reihenfolge der Beschreibungen entspricht dabei der Anordnung der Buttons von links nach rechts und von oben nach unten.

Button	Beschreibung
Quit	Das Programm wird verlassen. Etwaige Änderungen an der Video-Konfiguration werden nicht übernommen.
Apply	Der aktuelle Videomodus wird den eingestellten Werten angepaßt. Ist jetzt nur ein scharzes Bild zu sehen oder pfeift der Monitor, bricht man die Konfiguration mit [Alt]+[Strg]+[Backspace] ab. Etwaige Konfigurationsdateien, die zum Start des X-Window-Systems benötigt werden, werden dadurch aber nicht angepaßt!
Auto	Sollen die Änderungen, die Sie mit Hilfe der Buttons *Left*, *Right*, *Up*, *Down*, *Wider*, *Narrower*, *Shorter*, *Taller*, *Restore* und der speziellen S3-Buttons *InvertVCLK* und *EarlySC* durchgeführt wurden, sofort wirksam werden, muß man diesen Button anlicken. Dabei wird der jeweilige Zustand des Buttons beziehungsweise der zugehörigen Funktion umgeschaltet. Ist die Funktion aktiv, wird der Button invertiert dargestellt. Das Einstellen in dieser Betriebsart kann aber mitunter recht nervig werden, da nach jeder Änderung sofort die Ausgabe angepaßt wird. Tastet der Monitor dabei auch jedesmal das Bild dunkel, sieht man bei mehreren Tastendrücken einen flackernden Bildschirm.
Test	Damit wird getestet, ob der eingestellte Modus von der Grafikkarten-/Monitor-Kombination dargestellt werden kann. Dazu wird der Modus entsprechend umgeschaltet und man sollte zusätzlich ein Fenster mit einem Button auf dem neuen Desktop sehen. Nach kurzer Zeit (zirka fünf Sekunden) wird wieder in den alten Modus zurückgeschaltet, so daß man spätestens jetzt wieder ein Bild sehen sollte. Abgebrochen wird der Test übrigens durch Anklicken des Buttons *Abort Test Now*, aber dazu müßte man ja überhaupt etwas sehen. Ist dieser neue Modus in Ordnung, wird er durch Auswahl des Buttons *Apply* übernommen.
Restore	Damit werden die Einstellungen wiedergestellt, die beim Start des Programms *xvidtune* bestanden.
Fetch	Klickt man diesen Button an, werden beim X-Server die Informationen über die aktuellen Einstellungen abgefragt und entsprechend im Fenster angezeigt.
Show	Die aktuellen Einstellungen werden auf der Einheit *stdout* ausgegeben. Dies ist in der Regel die Konsole, von der aus *xvidtune* aufgerufen wurde. Die Ausgabe erfolgt im XF86Config-Modeline-Format.

Tabelle 12.7 (Fortsetzung auf nächster Seite)

Button	Beschreibung
Next	Der Videomodus wird entsprechend der in der Datei *XF86Config* enthaltenen Spezifikationen umgeschaltet. Die Datei befindet sich nach erfolgter Konfiguration im Verzeichnis »/usr/X11R6/lib/X11/«. Dabei wird der in der Datei nachfolgende Modus verwendet und die Änderung erfolgt sofort. Aber keine Sorge, man kann den Button *Prev* anklicken, um wieder in den alten Modus zu gelangen. Sollte der Button nicht sichtbar sein, wird der dargestellte Ausschnitt entsprechend verschoben, wenn man die Maus an eines der Ränder bewegt. Man kann sich das so vorstellen, daß der Desktop immer noch die vorige Größe besitzt, aber nur ein kleinerer Ausschnitt jeweils dargestellt wird. Startet man das X-Window-System später einmal direkt in diesem Modus, funktioniert das ganze mit dem Ausschnitt so natürlich nicht.
Prev	Der in der Datei angegebene vorherige Videomodus wird verwendet.

Tabelle 12.7: Die Funktion der Programmbuttons (links) in xvidtune

In der rechten Spalte sieht man unter der Button-Zeile die Ausgaben des Pixeltaktes (*Dot Clock*), der horizontalen und der vertikalen Synchronisationsfrequenz. Diese werden, wie bereits erwähnt, entsprechend den veränderten Einstellungen angepaßt. Damit wird auch kontrolliert, ob die Grafikkarten-/Bildschirm-Kombination den ergonomischen Anforderungen gerecht wird. Allerdings werden hier Phantasiewerte angezeigt, wenn man beispielsweise einen Framebuffer-Treiber einsetzt. Da dieser die Grafikhardware auf andere Weise ansteuert, sind die Werte nicht von Bedeutung.

Sind Sie Besitzer einer prähistorischen S3-Grafikkarte mit einem 964 oder 968 Chip (oder auch einem anderen S3-Chip ähnlichen Baudatums, wie beispielsweise einem 864), werden noch zusätzliche Einstellmöglichkeiten angeboten. Wird der Button *InvertVCLK* angeklickt, kann man den VCLK-Status von nicht invertiert zu invertiert umschalten. Nach der Umschaltung wird der Button revers dargestellt. Auch der EarlySC-Status wird umgeschaltet, wenn man den zugehörigen Button auswählt. Zusätzlich gibt es noch die Möglichkeit, BlankDelay-Werte einzustellen. Sie werden entweder direkt im Eingabefeld eingetragen oder der Vorgabewert wird mit Hilfe der Plus- und Minus-Button neben den Feldern eintragen. Normalerweise braucht man hier aber nichts einzustellen, die Vorgabewerte sind meist völlig ausreichend.

Sind die Änderungen ausgeführt, sollte man mindestens einmal den Button *Apply* betätigen, damit die Einstellungen zumindest für den Rest der Sitzung erhalten bleiben. Notieren Sie sich dann die Werte *für Pixel Clock, HDisplay, HSyncStart, HSyncEnd, HTotal, VDisplay, VSyncStart, VSyncEnd* und *VTotal*, sie werden später noch benötigt. Außerdem muß man sich diese Werte direkt nach dem Start von *xvidtune* aufschreiben, damit man die passende Zeile für die spätere Änderung finden kann.

Anschließend kann man das Programm mit *Quit* beenden.

Danach muß dann nur noch die Konfigurationsdatei mit einem Editor bearbeitet und die entsprechenden Werte eintragen werden.

12.7 Von Version 4 nicht unterstützte Karten

12.7.1 Framebuffer-Treiber

Wird Ihre Grafikkarte von XFree86 Version 4 nicht unterstützt, was häufig bei älteren Karten der Fall sein dürfte, können Sie stattdessen den Framebuffer-Treiber verwenden. Dieser bietet in der Regel eine Auflösung von 1024 x 768 Bildpunkten bei 16 Bit Farbtiefe, so daß mit ihm zumindest von den Werten her vernünftig zu arbeiten wäre. Auch auf Notebooks ist es mitunter die einzige Möglichkeit, einen Desktop sehen zu können. Eingerichtet wird der Treiber über das textbasierte Programm *xf86config*, da die grafische Einrichtung über *xf86cfg* nicht funktioniert.

Während der Konfiguration findet man bei der Wahl der Grafikkarte die passende Option gleich zu Anfang der Liste. Die weitere Einrichtung läuft dann so ab, als hätte man eine unterstützte Grafikkarte gewählt. Dabei sollte man auf jeden Fall als Farbtiefe den Wert 16 wählen. Beachten Sie, daß standardmäßig die kleinste Auflösung als erste probiert wird. Sollten Sie nach dem Starten des X-Window-Systems kein Bild sehen, können Sie die Auflösungen mit den Tastenkombination [Strg]+[Alt]+[Numerisch +] und [Strg]+[Alt]+[Numerisch -] umschalten. Auf dem Testsystem des Autors war nur bei einer Auflösung von 1024 x 768 Bildpunkten etwas zu erkennen, alle anderen waren unbrauchbar.

Leider liefert der Framebuffer-Treiber auf manchen Grafikkarten aber derart schlechte Bilder, daß an einen mehrstündigen Einsatz überhaupt nicht zu denken ist. Man kann zwar etwas erkennen, aber das Bild wirkt flau und nach kurzer Zeit bekommt man Kopfschmerzen. Aber immerhin, man sieht zumindest, was auf dem Desktop vor sich geht.

12.7.2 Server der Version 3.3.6

Aber es gibt noch eine Alternative, wenn man den Framebuffer-Treiber nicht verwenden will. Man setzt die alten X-Server der Version 3.3.6 ein, wobei aber das Grundsystem von der Version 4 stammt. Dazu fügt man das Paket *xservers-3.3.6* den anderen installierten Paketen hinzu. Dazu benötigt man aber zirka 40 MByte freien Platz im »/usr«-Dateisystem auf der Platte. Die Dateien des Pakets werden in das Verzeichnis »/usr/X11R6/X11R6.3« entpackt.

Steht der Platz im »/usr«-Dateisystem nicht zur Verfügung, kann man das Paket auch mit den Befehlen *bunzip2* und *tar* einzeln auspacken. Man kopiert Sie es dazu in ein Verzeichnis der Wahl, beispielsweise nach »/home/xserver«, und entpackt es dann mit

```
# tar -xzf xserver-3.3.6*
```

Die alten Server und die Maus

Wenn man die Server von XFree86 3.3.6 anstelle des neuen Servers einsetzt, wird man sich über die Mausgeschwindigkeit wundern: Es sind enorme Wege zurückzulegen, um den Cursor über den Bildschirm zu bewegen.

Der Grund für dieses Verhalten sind falsch gesetzte Mausparameter. Für solche Justierungsarbeiten steht das Programm *xset* zur Verfügung. Vernünftige zurückzulegende Wege erhält man mit einem Aufruf von *xset* mit der Parameterliste *m 10 1*:

```
xset m 10 1
```

»m« steht dabei für »Maus«, der erste Wert für die Beschleunigung und der zweite Wert für Geschwindigkeitumsetzung. Sie sollten im Zweifelsfalls auch andere Parameter in einer X-Konsole testen bis eine Mausgeschwindigkeit nach Ihrem Geschmack gefunden ist. Diese Angaben sind nicht permanent und gehen beim Neustart des Servers verloren. *xset* sollte deshalb in der passenden X-Startdatei mit den passenden Parametern aufgerufen werden, entweder also in der »~/.xinitrc«, »~/.xsession« oder auch der »/usr/X11R6/lib/X11/xdm/Xsession«.

Falls Sie die Mausdaten kaputtkonfiguriert haben und alles auf den Ursprungszustand zurücksetzen wollen verwenden Sie dazu den Aufruf

```
xset m default
```

Mit *xset* lassen sich auch andere XServer-Einstellungen steuern. Informationen dazu finden Sie auf der Man-Page des Programms.

Damit wird ein Unterverzeichnis mit dem Namen »X11R6.3« erzeugt, in dem sich die Dateien befinden. Beachten Sie, daß man in diesem Fall etwa doppelt so viel Platz benötigt, da zusätzlich zu den entpackten Dateien das tar-Archiv noch im Verzeichnis enthalten ist. Dafür braucht man den Speicherplatz nicht im »/usr«-Dateisystem, sondern in »/«. Außerdem müssen Sie im Unterverzeichnis »/usr/X11R6« noch einen Link mit dem Namen *X11R6.3* auf das korrekte Verzeichnis (hier: »/home/xserver/ X11R6.3«) erzeugen.

```
# ln -s /home/xserver/X11R6.3/ /usr/X11R6/X11R6.3
```

Nachdem das Paket installiert ist, muß man einen Link mit dem Namen *X* aus dem Verzeichnis »/usr/X11R6/bin« (und nicht »/usr/X11R6/X11R6.3«!) anlegen. Standardmäßig zeigt dieser auf die Datei »XFree86« im selben Verzeichnis. Den Link muß man so ändern, daß er auf den (alten) X-Server zeigt, der zu Ihrer Karte paßt, also beispielsweise auf »/usr/X11R6/X11R6.3/bin/XF86_S3«, wenn sich im Rechner eine S3-Karte befindet.

```
# ln -s /usr/X11R6/X11R6.3/bin/XF86_S3 /usr/X11R6/bin/X
```

Man kann sich das X-Server-Paket natürlich auch aus dem Package-Source-Tree generieren und installieren. Aber es ist sehr groß, da die gesamten Quellen von XFree86 3.3.6 heruntergeladen werden müssen. Im Normalfall ist also die Installation aus dem Binärpackage die bessere Lösung.

XFree86 3.3.6 statt 4.2

Falls Sie mit XFree86 4 überhaupt nicht zufrieden sind, besteht immer noch die Möglichkeit, von vorherein die Version 3.3.6 zu verwenden. Dazu laden Sie sich die Quellen von XFree86 3.3.6 entweder vom ftp-Server der XFree86.org oder der NetBSD.org (oder einem der Spiegelserver) herunter und entpacken es in ein Verzeichnis mit ausreichend freiem Platz. Dann führen Sie hier ein *make World* durch. Sie dürfen aber vorher auf keinen Fall die mitgelieferten X-Sets installiert haben.

Das Übersetzen des kompletten X-Pakets dauert recht lange. Das *make World* installiert die Programme und Daten aber auch gleich.

Sie sollten – wenn Sie so radikal vorgehen – nachher keine X-Programme aus Binärpaketen installieren, sondern sie aus den Quellen erzeugen, da einige Abhängigkeiten mit XFree86 3.3.6 anders gehandhabt werden müssen, beispielsweise wird bei der älteren Version kein FreeType-Paket mitgeliefert, es muß also beim Erzeugen der Pakete als Abhängigkeit kompiliert werden. Außerdem ist es dann möglich, aus dem Package-Source-Tree für die Konfiguration von X das alte Programm *XF86Setup* zu generieren (»usr/pkgsrc/x11/XF86Setup«).

Damit der Start tatsächlich klappt, benötigt man noch eine Konfigurationsdatei mit dem Namen »/etc/XF86Config«. Beachten Sie, daß Sie keine der Version 4 einsetzen können, da es erhebliche Unterschiede zwischen den beiden Versionen gibt. Sie können aber eine nehmen, die beispielsweise unter einem Linux-System mit XFree86 3.3.6 für diese Grafikkarte erzeugt wurde. Sollte der Austausch nicht über das Einbinden von Partitionen klappen, bietet sich dafür die gute alte Diskette an. Unter NetBSD muß man nur daran denken, den Typ des verwendeten Dateisystems explizit anzugeben, denn sonst sieht man nur eine Fehlermeldung, also beispielsweise

```
# mount -t msdos /dev/fd0a /mnt
```

Wird immer noch eine Fehlermeldung angezeigt, ist vermutlich die Diskette noch nicht eingelegt oder nicht lesbar.

Zweimal XF86Config

Es gibt kein Problem, wenn man versuchsweise den Server von XFree86 4 und einen von XFree86 3.3.6 verwenden will. Die Konfigurationsdateien sind zwar komplett verschieden befinden sich aber sinnvollerweise in unterschiedlichen Verzeichnissen. Während XFree86 4.2 alternativ entweder die Datei »/usr/X11R6/lib/X11/XF86Config« oder »/etc/X11/XF86Config« verwendet, laden die alten Server die Datei »/etc/XF86Config«. Man muß also nur den symbolischen Link »/usr/X11R6/bin/X« jeweils auf den anderen X-Server umhängen.

Anschließend kopieren Sie die Datei in das Verzeichnis »/etc«. Dort wird sie dann auf jeden Fall vom Server gefunden. Alternativ kann man auch eine Konfigurationsdatei erzeugen, aber es steht nur das textbasierte Werkzeug *xf86config* zur Verfügung. Außerdem muß man das richtige wählen, so daß folgender Befehl auszuführen ist:

```
# /usr/X11R6/X11R6.3/bin/xf86config
```

Dieses Verfahren ist allerdings sehr aufwendig, es ist empfehlenswert, sich von einem anderen Betriebssystem (auf demselben Rechner) eine vernünftig funktionierende Datei zu kopieren. Deren Inhalt könnte beispielsweise für eine alte S3-Karte wie folgt aussehen:

```
# See 'man XF86Config' for info on the format of this file

Section "Files"
  RgbPath   "/usr/X11R6/lib/X11/rgb"
  FontPath  "/usr/X11R6/lib/X11/fonts/local"
  FontPath  "/usr/X11R6/lib/X11/fonts/misc:unscaled"
  FontPath  "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
  FontPath  "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
  FontPath  "/usr/X11R6/lib/X11/fonts/Type1"
  FontPath  "/usr/X11R6/lib/X11/fonts/URW"
  FontPath  "/usr/X11R6/lib/X11/fonts/Speedo"
  FontPath  "/usr/X11R6/lib/X11/fonts/misc"
  FontPath  "/usr/X11R6/lib/X11/fonts/75dpi"
  FontPath  "/usr/X11R6/lib/X11/fonts/100dpi"
EndSection

Section "ServerFlags"
EndSection

Section "Keyboard"
   Protocol        "Standard"
   XkbRules        "xfree86"
   XkbModel        "pc105"
```

```
   XkbLayout       "de"
EndSection

Section "Pointer"            # Achtung:
   Protocol     "Microsoft"  # unter NetBSD praktisch immer "wsmouse"
   Device       "/dev/tty00" # unter NetBSD praktisch immer "/dev/wsmouse"
   BaudRate        1200
   Emulate3Timeout 50
   Resolution      100
   Buttons         3
   #Emulate3Buttons
EndSection

Section "Monitor"
   Identifier      "Primary Monitor"
   VendorName      "Unknown"
   ModelName       "Unknown"
   HorizSync       31.5-57.0
   VertRefresh     50-90
   Modeline "1024x768" 75.00 1024 1048 1184 1328 768 771 777 806 -hsync \
         -vsync # Aus drucktechnischen Gründen getrennt. Gehört zusammen!
EndSection

Section "Device"
   Identifier      "Primary Card"
   VendorName      "Unknown"
   BoardName       "S3 964 (generic)"
   Ramdac          "bt485"
   Chipset         "s3_generic"
   VideoRam        2048
# Use Option "nolinear" if the server doesn't start up
# correctly (this avoids the linear framebuffer probe).
# If that fails try option "nomemaccess".
# Refer to /usr/X11R6/lib/doc/README.S3, and the XF86_S3 man page.
EndSection

Section "Screen"
   Driver          "Accel"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 16
   SubSection "Display"
      Depth        8
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        15
```

```
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        16
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        24
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        32
      Modes        "1024x768"
   EndSubSection
EndSection

Section "Screen"
   Driver          "SVGA"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 16
   SubSection "Display"
      Depth        8
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        15
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        16
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        24
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        32
      Modes        "1024x768"
   EndSubSection
EndSection

Section "Screen"
   Driver          "VGA16"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
```

```
   SubSection "Display"
      Depth       4
      Modes       "1024x768"
   EndSubSection
EndSection

Section "Screen"
   Driver         "VGA2"
   Device         "Primary Card"
   Monitor        "Primary Monitor"
   SubSection "Display"
      Depth       1
      Modes       "1024x768"
   EndSubSection
EndSection

Section "Screen"
   Driver         "Mono"
   Device         "Primary Card"
   Monitor        "Primary Monitor"
   SubSection "Display"
      Depth       1
      Modes       "1024x768"
   EndSubSection
EndSection
```

Wie zu sehen ist, bestehen die Einträge aus sogenannten Sektionen, die wie nachfolgend beschrieben aufgebaut sind:

```
Section  "Sektionsname"
   Einträge
   ...
EndSection
```

Dabei beschreibt der Sektionsname, welche Aspekte bis zum Ende der jeweiligen Sektion (*EndSection*) eingestellt werden können. Daran kann sich eine weitere Sektion anschließen, die wieder mit dem Wort *Section* beginnt. Für den Sektionsnamen kann einer der in Tabelle 12.8 aufgeführten Oberbegriffe angegeben werden, wobei aber nicht unbedingt immer alle Sektionen enthalten sein müssen.

Der für das X-Window-System einzusetzende X-Server wird in der Sektion *Device* spezifiziert. Außerdem sind hier Einträge für den auf der Grafikkarte enthaltenen Speicher und die verwendeten Hardware-Bestandteile möglich.

In der Sektion *Files* werden die Pfadnamen der Verzeichnisse spezifiziert, in denen die Schriften enthalten sind. Außerdem befindet sich hier die Angabe des Pfads auf die RGB-Datenbank.

Zur Einstellung der Tastatur dient die Sektion *Keyboard*. Dazu gehören die Einstellung der Sprache, der Anzahl der Tasten und diverse weitere Optionen, die das Verhalten der Tastatur beeinflussen können.

Name	Kurzbeschreibung
Device	Gibt die Spezifikation der Grafikkarte an.
Files	Die Sektion enthält die Pfadnamen der benötigten Dateien.
Keyboard	Spezifiziert Optionen zur Konfiguration der Tastatur.
Module	Beschreibt das dynamische Laden der Module.
Monitor	Die Sektion enthält die Spezifikation des Monitors.
Pointer	In der Sektion wird die Konfiguration der Maus vorgenommen.
Screen	Beschreibt die Auflösung und Farbtiefe.
ServerFlags	Enthält spezielle Einstellungen für den Server.
XInput	In dieser Sektion befindet sich die Konfiguration erweiterter Eingabegeräte, wie beispielsweise Grafiktabletts oder Joysticks.

Tabelle 12.8: Die möglichen Sektionsnamen

Die Sektion *Module* gibt an, welche dynamischen Module geladen werden sollen. Allerdings wird sie meist nur im Zusammenhang mit erweiterten Eingabegeräten eingesetzt.

Die Konfiguration der Maus erfolgt in der Sektion *Pointers*. Dazu gehören unter anderem die Einstellung des Protokolls und die Angabe, über welches Gerät die Maus angesprochen wird. NetBSD besitzt mit seiner WS-Konsole eine Zwischenlayer, die die Low-Lovel-Devices der Maus auf das Device »/dev/wsmouse« abbildet. Deshalb ist dieses Gerät praktisch immer als Maus-Device anzugeben und eine USB-Maus funktioniert damit unter NetBSD genauso gut wie eine PS/2-Maus. Als Mausprotokoll ist dann *wsmouse* anzugeben.

Die Sektion *Monitor* legt die Daten des Bildschirms fest. Dabei werden sowohl die maximale als auch die minimale Horizontal- und Vertikalfrequenz angegeben. Außerdem ist in der Sektion eine Angabe über die für diesen Monitor zulässigen Videomodi enthalten. In ihr befindet sich auch die Modeline-Zeile, deren Modifikation vorher schon besprochen wurde. Die Zeile beginnt mit dem Wert der *Pixel Clock* (auch als *Dot Clock* bezeichnet) gefolgt von den Werten für die horizontale Display-Auflösung (entsprechend *HDisplay* bei *xvidtune*), dem Start- und Endpunkt des horizontalen Synchronisationssignals (*HSyncStart* und *HSyncEnd*) und der totalen Länge des Sync-Signals (*HTotal*). Daran schließen sich dann die Werte für die vertikalen Synchronisationssignale an, also vertikale Auflösung (*VDisplay*), Start- und Endpunkt (*VSyncStart* und *VSyncEnd*) und Gesamtlänge (*VTotal*). Ein solche Zeile sieht dann beispielsweise folgendermaßen aus:

```
ModeLine "640x480" 25.175 640 664 760 800 480 491 493 525
```

Wundern Sie sich aber nicht, wenn Sie eine solche Zeile nicht direkt finden. Die Daten können auch auf mehrere Zeilen aufgeteilt werden.

```
Mode "640x480"
    DotClock        25.175
    HTimings        640 664 760 800
    VTimings        480 491 493 525
EndMode
```

Die Zuordnung der Grafikkarte und des Monitors zu den möglichen Videomodi findet in der Sektion *Screen* statt. Außerdem sind hier Untersektionen möglich, die mit dem Wort *SubSection* beginnen und mit *EndSubSection* aufhören. Darin werden Optionen angegeben, die ebenfalls mit der Auflösung und Farbtiefe zusammenhängen.

In der Sektion *ServerFlags* kann man spezielle Optionen eintragen, die unter Umständen für einen reibungslosen Betrieb der Grafikkarte erforderlich sind. Sollen erweiterte Eingabegeräte verwendet werden, können diese in der Sektion *XInput* spezifiziert werden. Auch hier gibt es Untersektionen, mit deren Hilfe jeweils spezielle Geräte (wie beispielsweise Joysticks oder Grafiktabletts) konfiguriert werden.

12.7.3 Erster Start

Hat man also eine passende Datei zur Verfügung, kann man das System bereits mit *startx* starten, was ohne größere Schwierigkeiten funktionieren sollte. Man sieht den Desktop und auch die Schriften werden angezeigt. Dennoch ist es besser, den bei der Installation mit *pkg_add* angezeigten Hinweis zu beachten, weil der alte Server mit den neuen Schriften nicht vernünftig umgehen kann. Das bedeutet, daß man die Datei »/etc/XF86Config« in einen Editor laden sollte und die Pfade für die Schriftdateien anpassen muß. Die Pfadangabe »/usr/X11R6/« ist durch »/usr/X11R6/X11R6.3/« zu ersetzen.

```
FontPath   "/usr/X11R6/X11R6.3/lib/X11/fonts/Type1"
FontPath   "/usr/X11R6/X11R6.3/lib/X11/fonts/Speedo"
FontPath   "/usr/X11R6/X11R6.3/lib/X11/fonts/misc"
FontPath   "/usr/X11R6/X11R6.3/lib/X11/fonts/75dpi"
FontPath   "/usr/X11R6/X11R6.3/lib/X11/fonts/100dpi"
```

Damit die Änderungen wirksam werden, ist ein Neustart des X-Servers erforderlich. Allerdings sieht der Desktop ähnlich aus, was aber auch kein Wunder ist, da sich die Ausgestaltung der Schriften nicht geändert hat. Das System sollte aber mit den alten Schriften stabiler laufen und weniger Core-Dumps produzieren.

12.8 Die mitgelieferten Fenstermanager

Dieses Kapitel ist eine Kurzbeschreibung der wichtigsten im NetBSD-System mitgelieferten Fenstermanager. Sie müssen selbst entscheiden, welcher von den angebotenen eingesetzt werden soll. Beachten Sie, daß bei den Abhängigkeiten nur die des Fenstermanager-Pakets aufgeführt sind. Mitunter gibt es noch weitere Abhängigkeiten für die hier aufgelisteten Pakete.

12.8.1 Afterstep

Name:	afterstep-1.8.11nb1	
Abhängigkeiten:	asclock	Uhr für das NextStep-System, die Datum-/Zeitanwendungen emulieren kann.
	gsm	Konvertierungsroutinen für u-law-Audio in das GSM-Format.
	jpeg	Funktionen zum Bearbeiten und Anzeigen von JPEG-Dateien.
	png	Bibliothek zum Bearbeiten und Anzeigen von PNG-Dateien.
	rplay	Programm zum Abspielen und Aufnehmen von Sound-Dateien verschiedenster Formate.
	rx	Schnittstelle nach POSIX-Norm zur Programmierung mit regulären Ausrücken.
	xp	Anzeige von Dateien.
Beschreibung:	Emuliert das Aussehen und die Bedienung von NextStep. Er ist hoch konfigurierbar und kann für den Gnome-Desktop eingesetzt werden. Afterstep wird über die Zeile `exec afterstep` gestartet, die beispielsweise in die Datei ».xinitrc« eingetragen wird (als einzige Zeile). Damit wird er bei jedem Aufruf hochgefahren. Beachten Sie in diesem Zusammenhang, daß sich das Format einiger Konfigurationsdateien geändert hat. Wie sich das auswirkt, entnehmen Sie der Online-Dokumentation.	

12.8.2 blackbox

Name:	blackbox-0.62.1
Abhängigkeiten:	–
Beschreibung:	Ein Fenstermanager, der in C++ geschrieben wurde. Er ist klein, schnell und unterstützt mehrere Desktop-Umgebungen. blackbox läßt sich beispielsweise mit dem Skript *startx* über den Eintrag *exec blackbox* in der Datei ».xinitrc« im Heimatverzeichnis hochfahren.

Abb. 12.18: Der Bildschirm von »blackbox« ist sehr aufgeräumt

12.8.3 9wm

Name:	9wm-1.1
Abhängigkeiten:	–
Beschreibung:	Ein Fenstermanager, der versucht, den Plan9-Fenstermanager 8-1/2 für das X-Window-System zu emulieren. Er bietet eine einfache, aber komfortable Benutzerschnittstelle ohne großen Schnickschnack.

12.8.4 Enlightenment

Name:	enlightenment-0.16.4nb1	
Abhängigkeiten:	esound	Enlightened Sound Dämon
	fnlib	24-Bit Farbausgabe-Unterstützung für X.
	freetype-lib	Schriften-Manager-Bibliothek für TrueType-Schriften.
	glib	Nützliche Routinen für die C-Programmierung.
	gtk+	Werkzeug zum Erzeugen von X11-Benutzerschnittstellen.
	imlib	Bildbearbeitungsbibliothek für X11.
	jpeg	Funktionen zum Bearbeiten und Anzeigen von JPEG-Dateien.
	libaudiofile	Soundbibliothek für SGI-Audiodateien.
	libghttp	Gnome http-Client-Bibliothek.
	libungif	Bibliothek zur Bearbeitung von GIF-Dateien.
	netpbm	Werkzeug zur Konvertierung von Grafikdateien.
	perl	Die Skriptsprache Perl. Benötigt wird das Paket 5.6.1nb7 (Meta-Datei zur Komplettinstallation).
	png	Bibliothek zum Bearbeiten und Anzeigen von PNG-Dateien.
	pth	Portable GNU Thread Bibliothek.
	tiff	Bibliothek zum Bearbeiten von TIFF-Dateien.
Beschreibung:	Ein weiterer Fenstermanager, der hochkonfigurabel ist und der auf ein besonderes grafisches Aussehen hin ausgelegt ist.	

12.8.5 sawfish

Name:	sawfish-1.0.1nb2	
Abhängigkeiten:	control-center	Werkzeug zur einfachen Konfiguration der Gnome-Umgebung.
	db3	Berkeley Datenbank-System.
	esound	Enlightened Sound Dämon.
	GConf	Datenbank-System für Konfigurationseinträge des Gnome-Desktops.
	gdbm	Funktionen für eine hashed Datenbank.
	gdk-pixbuf	Anzeige-Bibliothek für Bilddaten.
	gdk-pixbuf-gnome	Anzeige-Bibliothek für Bilddaten für den Gnome-Desktop.
	gindent	Formatierungswerkzeug für C-Dateien.
	glib	Routinen für die C-Programmierung.
	gmp	Arithmetische Berechnungsbibliothek.
	gnome-libs	Gnome-Bibliotheken.
	gnome-vfs	Virtuelles Gnome-Dateisystem.
	gtk+	Werkzeug zum Erzeugen von X11-

		Benutzerschnittstellen.
	imlib	Bildbearbeitungsbibliothek für X11.
	jpeg	Funktionen zum Bearbeiten und Anzeigen von JPEG-Dateien.
	libaudiofile	Soundbibliothek für SGI-Audiodateien.
	libglade	Laufzeit-Interpreter für GLADE-GUI-Dateien.
	librep	Emacs-Lisp-Laufzeitbibliothek.
	libungif	Bibliothek zur Bearbeitung von GIF-Bildern.
	libxml	Parser für XML-Dateien.
	netpbm	Werkzeug zur Konvertierung von Grafiken.
	oaf	Objekt-Aktivierungs-Framework für Gnome.
	ORBit	CORBA ORB für C.
	perl	Die Skriptsprache Perl.
	png	Bibliothek zum Bearbeiten und Anzeigen von PNG-Bildern.
	popt	Bibliothek zum Parsen der Kommandozeile.
	pth	Portable GNU Thread Bibliothek.
	readline	Bibliothek, um gerade getätigte Eingaben erneut nutzen und bearbeiten zu können.
	rep-gtk	Emacs-Lisp-Laufzeitbibliothek-Bindungen (librep) für Gnome.
	sawfish-replibs	librep-Lisp-Bibliotheken.
	scrollkeeper	Katalog-System für Dokumentationen.
	tiff	Bibliothek zum Bearbeiten von TIFF-Dateien.
Beschreibung:	Sehr großer Fenster-Manager, der mit einer Lisp-ähnlichen Skriptsprache gesteuert werden kann. Fast alle Bestandteile des Systems können konfiguriert werden. Außerdem ist er einer der Fenstermanager, die gut mit dem Gnome-Desktop zusammenarbeiten.	

12.8.6 WindowMaker

Name:	windowmaker-0.80.1	
Abhängigkeiten:	Hermes	Grafikbibliothek für Spiele.
	jpeg	Funktionen zum Bearbeiten und Anzeigen von JPEGs.
	libungif	Bibliothek zur Bearbeitung von GIF-Dateien.
	png	Bibliothek zum Bearbeiten und Anzeigen von PNGs.
	tiff	Bibliothek zum Bearbeiten von TIFF-Dateien.
Beschreibung:	Fenstermanager, der die Bedienung, das Aussehen und das Verhalten von Nextstep nachbilden soll. Er ist relativ klein, schnell und bietet eine Reihe von einfach auszuführenden Konfigurationsmöglichkeiten. *WindowMaker* kann auch mit dem Gnome-Desktop verwendet werden.	

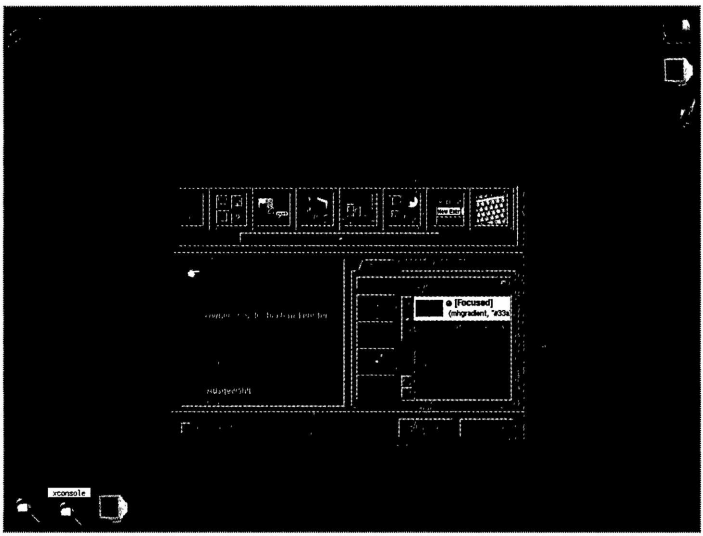


Abb. 12.19: Der Desktop von WindowMaker mit dem Konfigurationsmenü

12.9 Weitere Fenstermanager

Zusätzlich zu den genannten gibt es noch eine Reihe weiterer teilweise gebräuchlicher Fenstermanager, die an dieser Stelle nur kurz vorgestellt werden sollen:

Manager	Beschreibung
amiwm	Emuliert das Aussehen und die Funktionalität der Amiga Workbench.
ctwm	Eine Erweiterung zum standardmäßig eingerichteten Fenstermanager *twm*, die virtuelle Bildschirme unterstützt. Außerdem werden noch weitere Funktionen nachgerüstet, die beispielsweise das Aussehen der Fenster verbessern. *ctwm* läßt sich einfach statt des *twm* starten, so daß in den Startdateien vor dem *twm* nur ein c hinzugefügt werden muß.
flwm	Kleiner und schneller Fenstermanager, der ohne Icons auskommt.
fvwm1	Bietet 3D-Aussehen für Windows und mehrere/virtuelle Desktops. Mit Hilfe einer Buttonleiste können die wichtigsten Funkionen aufgerufen werden. Benötigt nur die Hälfte des Platzes von *fvwm2*. Beachten Sie, daß *fvwm1* und *fvwm2* einige Dateien gemeinsam benutzen. Setzen Sie beide ein und löschen Sie dann einen von beiden, kann unter Umständen ein benötigtes Paket mit gelöscht werden.

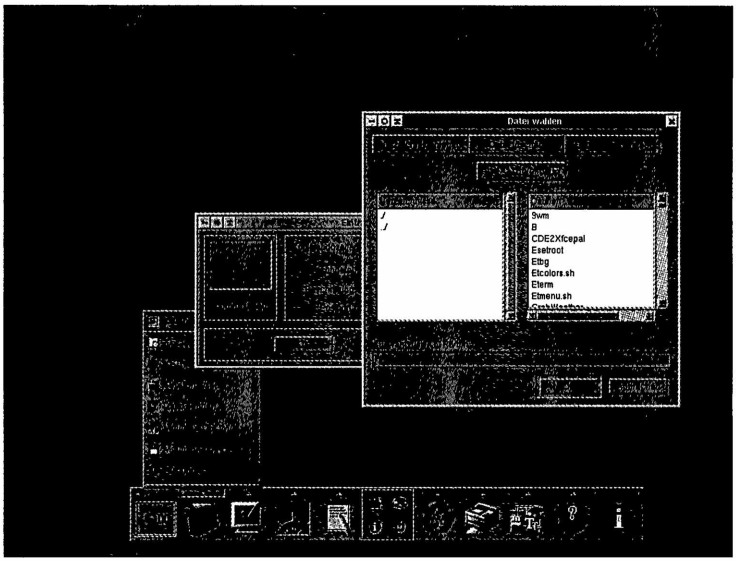


Abb. 12.20: XFCE ist sehr aufgeräumt und kommt Anwendern kommerzieller Unix-Systeme gleich bekannt vor

fvwm2	Bietet 3D-Aussehen für Windows und mehrere/virtuelle Desktops sowie einige Erweiterungen gegenüber dem *fvwm1*. Mit Hilfe einer Buttonleiste können die wichtigsten Funktionen aufgerufen werden. Der Fenster-Manager ist hoch konfigurierbar. Beachten Sie, daß *fvwm1* und *fvwm2* einige Dateien gemeinsam benutzen. Setzen Sie beide ein und löschen Sie dann einen von beiden, kann unter Umständen ein benötigtes Paket mit gelöscht werden.
fvwm95	Anpassung von *fvwm2* an das Aussehen von Windows95. Damit können Einsteiger in Ihrer vertrauten Umgebung weiterarbeiten.
icewm	Schneller und kleiner Fenstermanager, der das Aussehen von Motif, OS/2 Warp und Windows 95 emulieren kann. Alle Funktionen können im Prinzip sowohl über die Tastatur als über die Maus ausgeführt werden. Allerdings ist er weniger konfigurierbar als beispielsweise *fvwm*.
ion	Fenstermanager, der den Bildschirm in Rahmen statt in überlappende Fenster teilt, wie dies beispielsweise auch beim *Emacs* der Fall ist. Das Umschalten zwischen den einzelnen Tasks kann mit der Tastatur erfolgen, so daß keine Maus vonnöten ist.

lwm	Fenstermanager, der keine Icons, Buttonleisten oder ähnliches bietet. Dafür ist er sehr klein und belegt wenig Platz, kann aber nicht konfiguriert werden.
olvwm	OpenLook Virtual Window Manager, der aus dem *olwm* hervorgegangen ist und mit dem XView-Toolkit eingesetzt wird. Er kann virtuelle Bildschirme verwalten.
xfce	XFCE ist ein Clone der CDE. Dieser Window Manager ist im Package-Source-Tree aber nicht bei den Window Managern, sondern in der generischen Rubrik »x11« zu finden. Einen Eindruck vermittelt Abbildung 12.20

12.10 KDE 3 – K Desktop Environment

Sie werden sich darüber wundern, daß die Beschreibung des KDE-Systems[1] nicht im vorigen Kapitel gegeben wurde. Das liegt daran, daß der Begriff KDE nicht einen Fenstermanager an sich beschreibt, sondern einen ganzen Desktop, für den auch gleich ein Fenstermanager mitgeliefert wird. Dieser wird aber selten durch einen anderen ersetzt, so daß dem KDE-System dieses eigene Kapitel gewidmet ist.

12.10.1 Einrichtung

Da das KDE-System standardmäßig nicht mit eingerichtet wird, muß man es erst nachinstallieren. Dabei ist es aber nicht erforderlich, alle Bestandteile mühsam per Hand einzeln einzurichten, sondern es gibt ein übergeordnetes Paket, das das erledigt. Die Installation selbst führt die Funktion *pkg_add* aus, wobei in der Regel nicht überprüft wird, ob noch genügend Platz im Dateisystem vorhanden ist. Probleme bei der Installation können also erst dann auftreten, wenn während des Kopierens der Dateien der freie Platz knapp wird. Dabei ist teilweise auch nur ein Dateisystem betroffen, beispielsweise das separate »/usr«-System. Besonders ärgerlich ist das, wenn beim zu installierenden Paket weitere Unterpakete aufgerufen werden, wie das bei dem Paket *kde-3.0.3*, mit dem das komplette KDE-System eingerichtet wird, der Fall ist. Dieses ist nur drei Kbyte groß, ruft aber (unter anderem) alle anderen KDE-Pakete auf, die dann nacheinander installiert werden. Damit ist die Bestimmung der realen Installationsgröße mühselig.

---

[1] An dieser Stelle wird das aktuelle KDE 3 besprochen. Die Packages enthalten bei NetBSD aber außerdem auch die alten Versionen 1.1.2 und 2.2.2. Auf schnellen Rechnern ist KDE 3 sicher die bessere Wahl, auf alten Maschinen stößt man bei den neuen ressourcenverschlingenden Versionen aber schnell an die Grenzen der Hardware. Bei einer Maschine mit weniger als 200 Mbyte RAM beginnt KDE 3 bereits ohne das Starten einer Anwendung Speicher auszulagern. KDE 1 kommt dagegen auch auf Maschinen mit weniger als 32 Mbyte RAM zurecht. KDE 2 gibt sich mit weniger als 128 Mbyte RAM zufrieden. Bei KDE 1 ist im Gegensatz zu KDE 2 und KDE 3 auch immer die deutsche Sprachunterstützung enthalten. Sie gehört hier zum Grundsystem.

Erscheint bei der Installation die Fehlermeldung, daß das Dateisystem voll ist, kann man ohne Probleme mittels der Tastenkombination [Strg]+[C] abbrechen. Doch leider wird dabei das installierte Paket nicht in die Paketdatenbank übernommen, so daß man gegebenenfalls die bereits kopierten Dateien von Hand löschen muß. Doch welche sind das? Im vorigen Beispiel ist das Ganze noch etwas verzwickter, da ja noch weitere Pakete betroffen sind, so daß man erst deren Dateinamen bestimmen muß. Außerdem weiß man dann nicht, von welchem Paket einzelne Dateien noch auf der Platte vorhanden sind, so daß die Suche etwas komplizierter wird.

Um diese mühselige Vorgehensweise zu ersparen, wird nachfolgend ein Verfahren vorgestellt, das die Angelegenheit etwas bequemer macht. Dazu muß aber erst einmal bestimmt werden, welche Pakete in der Datenbank bereits aufgelistet worden sind. Dies erledigt einfachsten der Befehl

```
# pkg_info -a
```

Mit ihm werden alle Pakete mit einer kurzen Beschreibung angezeigt, so daß man sich schon mal orientieren kann. Für das obige Beispiel könnte folgende Zeile angezeigt werden:

```
qt3-libs-3.0.5  C++ X GUI toolkit
```

Es ist zu sehen, daß die QT-Bibliothek der Version 3 installiert ist, die zwingend für den Betrieb des KDE-Desktops erforderlich ist. Will man noch wissen, wieviel Platz dieses Paket benötigt, geht das mit folgendem Aufruf:

```
# pkg_info -s qt3-libs-3.0.5
```

Man erkennt daran, daß durch die Deinstallation 25932856 Byte freiwerden würden. Sie erfolgt durch Eingabe des Befehls

```
# pkg_delete qt3-libs-3.0.5
```

Bestehen keine weiteren Abhängigkeiten, wird das Paket jetzt gelöscht, ansonsten kommt eine Fehlermeldung, die mitteilt, welches Paket vorher gelöscht werden sollte. Leider muß zum Löschen eines Pakets immer der komplette(!) Dateiname angegeben werden, Platzhalter sind nicht erlaubt.

Wenn man jetzt noch weiß, welche Pakete beispielsweise zu dem vorher angesprochenen KDE-Paket gehören, könnte man zumindest schon mal ein wenig Platz auf der Platte schaffen. Doch alle Dateien, die zu keinem eingetragenen Paket gehören, findet man auf diese Weise nicht.

Das geht nur, wenn man das Paket installiert und deinstalliert, zu dem sie gehören. Wenn man nur ein Paket installiert, hat man natürlich keine Probleme mit der Zuordnung. Doch Sie haben keinen ausreichenden Speicherplatz, denn sonst wäre es nicht zu

dieser Situation gekommen. Sie müssen also vorher andere Pakete löschen, um das gewünschte Paket installieren zu können. Welche dafür in Frage kommen, findet man beispielsweise mit dem Befehl

```
# pkg_info -as
```

heraus. Neben dem Namen wird zusätzlich die Größe des Pakets angezeigt, leider fehlt aber die Beschreibung. Beste Kandidaten zu diesem Zweck sind erstmal die japanischen Vektor-Schriften (*ja-vflib-lib-2.24.2* und *watanabe-vfont-19930318*), die vermutlich sowieso nicht benötigt werden. Zusätzlich kann man noch weitere Pakete auswählen, die man aktuell nicht benötigt und die man jetzt deinstalliert. Nachdem dann das andere Paket installiert und wieder deinstalliert wurde, sollte mehr Platz auf der Platte vorhanden sein. Allerdings kann es erforderlich sein, die Überprüfung der Abhängigkeiten zu deaktivieren, was mit der Option *-f* passiert. Da das gerade installierte Paket sofort wieder deinstalliert wird, führt diese Vorgehensweise zu keinen Problemen im restlichen System.

Danach kann man daran gehen, die für dieses Verfahren deinstallierten Pakete wieder zu installieren, bis das System wieder an dem Punkt angekommen ist, an dem es auch zu dem Zeitpunkt war, als die fehlgeschlagene Installation gestartet wurde.

Doch noch einmal zurück zum Paket *kde-3.0.3*. Entpackt man es, findet man eine Datei mit dem Namen *+SIZE_ALL*. In ihr ist die Gesamtgröße aller zu installierenden Pakete enthalten. Allerdings bietet diese nur einen Anhaltspunkt, da nicht bekannt ist, wie sich die Byte auf die normalerweise vorhandenen Dateisysteme »/usr« und »/« aufteilen. Selbst wenn eine Überprüfung mit diesem Wert stattfinden würde, könnte es immer noch sein, daß eines der Dateisysteme überlaufen würde, so daß sie keine Aussage liefern würde.

Zusätzlich ist in dem Paket eine Datei mit dem Namen »+CONTENTS« vorhanden, in der beschrieben wird, welche Abhängigkeiten es gibt. Man entnimmt ihr, welche Pakete installiert werden sollen, so daß man nur den Inhalt auswerten muß, um die benötigten Paketnamen zu erhalten.

Beachten Sie in diesem Zusammenhang aber, daß einige erforderliche Pakete auch anderweitig genutzt werden, so daß Sie sie nicht deinstallieren dürfen. Da aber die Abhängigkeiten überprüft werden, dürfte das kein Problem darstellen.

Wenn Ihnen dieses Problem beim erstmaligen Einrichten des Systems begegnet, sollten Sie die Installation wiederholen, diesmal aber mit einer anderen Aufteilung der Dateisysteme »/usr« und »/«. Sonst kommen Sie irgendwann wieder an den Punkt, an dem das gleiche Problem auftritt und Sie haben nichts gewonnen. Zusätzlich wäre es sicherlich wünschenswert, die NetBSD-Partition insgesamt zu vergrößern, so daß mehr Platz zur Verfügung steht. Gleichzeitig sollten Sie aber auch die automatische Aufteilung der Disklabel ändern, so daß auf dem »/usr«-Bereich mehr Platz zur Verfügung steht. Weitere Informationen zur Paketverwaltung und den dazu erforderlichen Befehlen finden Sie auf den Manual-Seiten von *pkg_add*, *pkg_delete* und *pkg_info*.

Ist dagegen genügend Platz in den Unterpartitionen, werden alle benötigten Pakete installiert und man braucht sich im Prinzip um nichts mehr groß zu kümmern, zumindest was die Einrichtung betrifft. Was aber noch ansteht ist die Anpassung der Konfigurationsdateien, damit das System überhaupt gestartet werden kann.

12.10.2 Konfiguration und Start

Nachdem die Pakete installiert sind, kann man daran gehen, den KDE-Desktop zu starten. Dazu erzeugt man im Heimatverzeichnis eine leere Datei mit dem Namen ».xinitrc« und trägt die nachfolgende Zeile darin ein:

```
startkde
```

Sollte das nicht funktionieren, geht es auch mit

```
exec startkde
```

Nun gibt man in einer Textkonsole den Befehl

```
# startx
```

ein, wodurch nach geraumer Zeit der Desktop zu sehen sein sollte. Er ist ansonsten über die Tastenkombination [Alt]+[Strg]+[F5] zu erreichen, mit [Alt]+[Strg]+[F1] bis [Alt]+[Strg]+[F4] kann man auf die Textkonsolen wechseln.

Wie man an der Oberfläche erkennt, werden alle Menüs und Punkte in Englisch angezeigt. Will man die deutsche Unterstützung, muß sie explizit nachinstalliert werden. Das ist nicht so einfach, weil primär NetBSD keine Packages für die i18n-Unterstützung von KDE enthält. Um diese Packages zu erhalten, können Sie NetBSD 1.6 und KDE 3.0.3 von der CD-ROM der Ausgabe 6'02 der Zeitschrift freeX installieren oder Sie bauen sich ein eigenes Package wie in Kapitel 5 dieses Buchs beschrieben. Vielleicht erbarmt sich aber jemand im Packages-Team der NetBSD-Gruppe und führt entsprechende Packages offiziell in das System ein.

Ist die Installation – egal woher das Paket bezogen wurde – erfolgreich verlaufen, muß man die Einstellungen noch ändern. Das geht über das Control Center, in dem der Punkt *Personalization/Country & Language* zu wählen ist. Im Feld *Country* wählt man den Eintrag *Europe, Central/Germany*, wodurch im Feld *Languages* die entsprechenden Sprachen angepaßt werden. Standardmäßig befindet sich vermutlich immer noch der Eintrag *English US*, so daß man mit *Add Language* eine Sprache hinzufügen muß. Sollte in der Menüstruktur, die durch Anklicken des Buttons aufklappt, der Punkt *German* nicht zu sehen sein, ist er unter *Other* verfügbar. Wählt man ihn aus, wird er unter *Languages* angezeigt. Damit die Änderungen wirksam werden können, muß man den Button *Apply* anklicken. Danach erscheint eine Warnung, daß sich die Änderungen nur auf neu gestartete Programme auswirken. Für eine globale Wirksamkeit muß man sich erst ab- und wieder anmelden.

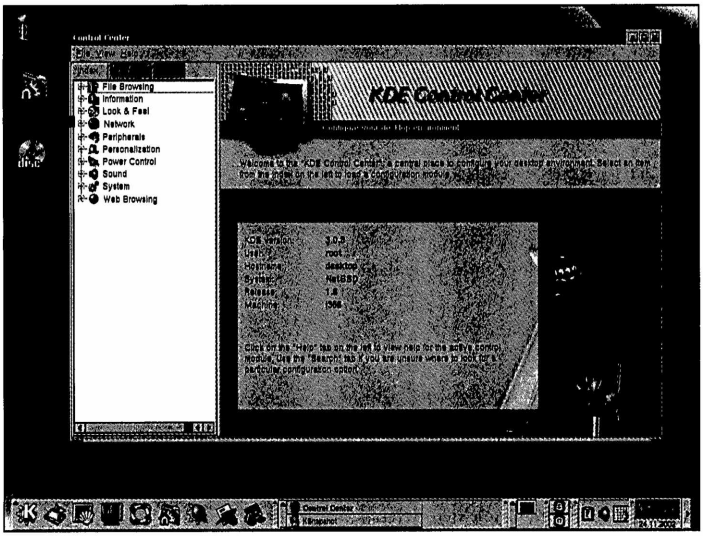


Abb. 12.21: Der KDE-3-Desktop unter NetBSD/i386

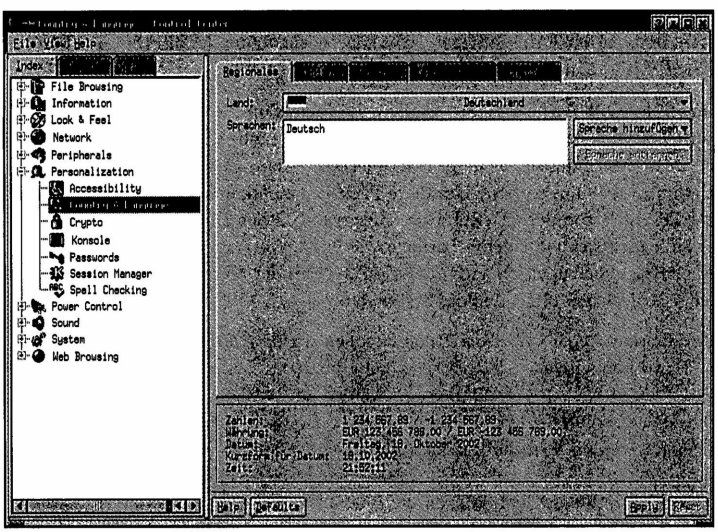


Abb. 12.22: Die Ländereinstellungen im KDE Control Center. Die i18n-Erweiterungen gehören nicht zum Standard-Lieferumfang von KDE bei NetBSD und müssen selbst generiert werden

Die weitere Anpassung des Desktops an die jeweiligen Wünsche erfolgt am einfachsten mit den weiteren Funktionen im Control Center beziehungsweise Kontrollzentrum. Mit diesem Werkzeug muß man nicht mehr direkt einzelne Textdateien ändern, die sonst erst mühsam gesucht werden müssen.

Sollte es dennoch einmal notwendig sein, findet man die zugehörigen Konfigurationsdateien im ».kde«-Unterverzeichnis im Heimatverzeichnis. Dabei werden hier aber nur die Einstellungen abgelegt, die den Standard überschreiben. Die für den Desktop sind beispielsweise in der Datei »kdesktoprc« gespeichert, die sich im Verzeichnis »$HOME/.kde/share/kde/config« befindet. Die Standardeinstellungen einzelner Programme sind in Dateien im Verzeichnis »/usr/X11R6/share/kde/config« gesichert. Dort kann man Änderungen vornehmen, wenn sie für alle Benutzer des Systems gleichermaßen gelten sollen, obwohl sie sich lokal natürlich wieder überschreiben lassen. Da es sich in allen Fällen um Textdateien handelt, sind Anpassungen mit einem Texteditor problemlos möglich.

12.10.3 Minimalsystem

Wenn man nur eine kleine Partition (kleiner 800 MByte) für NetBSD reserviert hat und dennoch das KDE-System nutzen will, darf auf keinen Fall das KDE-Meta-Paket für die Einrichtung verwendet werden. Stattdessen müssen die erforderlichen Bestandteile von Hand eingerichtet werden, was zwar mühsam ist, aber funktioniert.

Allerdings müssen zu diesem Zweck auch die Unterpartitionen entsprechend groß sein, denn sonst läuft eine davon unter Umständen über. Speziell benötigt man dafür mindestens zirka 100 MByte Platz auf der »/«-Unterpartition, da die Pakete erst in einem temporären Verzeichnis entpackt werden, bevor die einzelnen Dateien in die zugehörigen Verzeichnisse kopiert werden. Hat man diesen Platz nicht, läßt sich das Paket nicht installieren.

Selbst bei einer ein Gbyte großen Partition treten Probleme bei der Einrichtung auf, wenn man das Meta-Paket nimmt und die Standardeinteilung der Unterpartitionen übernimmt. In diesem Fall ist die »/«-Unterpartition zirka 145 MByte groß, die Swap-Partition zirka 250 MByte und die »/usr«-Unterpartition 690 MByte. Nach der Installation verbleiben in »/« noch zirka 97 MByte und in »/usr« zirka 55 MByte. Das Problem bei der Installation besteht wieder darin, daß zwischenzeitlich im »/«-Dateisystem kein Platz mehr vorhanden ist und damit Abhängigkeiten nicht aufgelöst werden können. Letztendlich führt das hier aber zu keinen allzu großen Schwierigkeiten, da die abhängigen Pakete später sowieso installiert werden.

Wenn man NetBSD auf einem System mit so wenig(?!) Plattenplatz einrichtet, muß man unter Umständen daran denken, die Aufteilung der Gesamtpartition entsprechend zu ändern, denn sonst läuft sehr schnell das »/usr«-Dateisystem über, wenn man noch weitere Programme installieren will. In diesem Fall sollte man die automatische Aufteilung so ändern, daß der Swap-Bereich kleiner wird. Für ihn reicht es häufig aus, die doppelte Größe des Hauptspeichers zu wählen, so daß man dort sparen kann. Den auf diese Weise verfügbaren Platz verteilt man am besten zum größten Teil auf das »/usr«-Dateisystem und der Rest wird dem »/«-Dateisystem zugeschlagen. Das installierte KDE-System läuft auf diese Weise ohne Probleme, nur ist natürlich sehr wenig Platz

für Erweiterungen vorhanden. Aus diesem Grund wäre es dann ratsam, die Partition doch zu vergrößern, sofern dies möglich ist. Das ist sicherlich auch erforderlich, wenn man noch mehrere Pakete zum Arbeiten einrichten will. Generell gilt also, daß die gesamte Partition größer als ein Gbyte sein sollte, wenn man XFree86 4 und den KDE-3-Desktop einsetzen will. Kleinere Partitionen bieten schon nicht mehr genügend Platz, um damit später vernünftig arbeiten zu können.

Doch nochmal zurück zur Installation des Minimalsystems. Wie bereits zu Anfang erwähnt, muß man die erforderlichen Pakete einzeln einrichten, damit auch nur wirklich das auf die Platte kommt, was man zu diesem Zweck benötigt. Als werden die von der QT-Bibliothek benötigten Pakete *glut*, *jpeg*, *lcms*, *mng*, *png* und *pth* installiert. Dabei kann man auch das Platzhalterzeichen »*« einsetzen, um nicht jedesmal den kompletten Namen eingeben zu müssen. Damit man bei Mehrdeutigkeiten nur ein Paket installiert, gibt man nach dem eigentlichen Namen ein Minuszeichen an.

```
# pkg_add glut-*
```

Sollte man dabei eine Fehlermeldung sehen, daß doch noch eine Abhängigkeit gegeben ist, dreht man die Reihenfolge der Pakete um beziehungsweise installiert zuerst das, dessen Namen man in der Fehlermeldung sieht.

Danach kann man das Paket *qt3-libs* installieren. Dies dauert eine geraume Zeit, sollte nun aber keine Probleme machen, da alle Abhängigkeiten aufgelöst sind. Allerdings setzt das voraus, daß man genügend Platz auf der Festplatte zur Verfügung hat.

Anschließend kommt als erstes das Paket mit den Bibliotheken des KDE-Systems an die Reihe, *kdelibs*. Für diese müssen zusätzlich noch die folgenden Pakete installiert werden, damit man keine Fehlermeldung angezeigt bekommt: *arts*, *libaudiofile*, *libiconv*, *libxml2*, *libxslt*, *pcre*, *perl* und *tiff*. Erst danach sollte die *kdelibs* eingerichtet werden.

Womit wir auch schon zum letzten Paket des KDE-Systems kommen, *kdebase*. Davor ist es aber erforderlich, noch die Pakete *openldap* und *qt3-tools* einzurichten, erst danach ist *kdebase* an der Reihe. Ist auch dieses installiert, funktioniert der Desktop schon, nur sind jetzt erst wenige KDE-Programme eingerichtet. Aber immerhin kann man das System zu diesem Zeitpunkt schon konfigurieren und benutzen.

12.11 Gnome

Auch das Gnome-System fehlt in der Liste mit den Fenstermanagern. Das ist aber auch kein Wunder, denn für Gnome benötigt man explizit einen Fenstermanager aus der Liste, der mit dem Desktop etwas anfangen kann, wie beispielsweise *sawfish*.

12.11.1 Einrichtung

Ähnlich wie für das KDE-System gibt es auch für Gnome ein Meta-Paket, das die für den Desktop erforderlichen Pakete automatisch einrichtet, so daß man sich um nichts mehr kümmern muß. Das setzt aber wieder voraus, daß genügend Platz in den Unterpartitionen vorhanden ist. Leider wird auch bei diesem Paket nicht überprüft, ob der

vorhandene Platz ausreicht, so daß unter Umständen wieder ein Verfahren angewandt werden muß wie im KDE-Kapitel beschrieben.

12.11.2 Konfiguration und Start

Damit das System startet, muß man wieder die Konfigurationsdateien im Heimatverzeichnis anpassen. Wenn man das System per Befehl *startx* hochfahren will, ist die Startdatei ».xinitrc« zu verändern. Man trägt folgendes in die Datei ein:

```
exec /usr/X11R6/etc/gdm/gnomerc
```

Alternativ ist auch die Anweisung

```
exec /usr/X11R6/etc/gnomerc
```

möglich. Gibt man jetzt an einer Kommandozeile den Befehl *startx* an, sollte man den Inhalt von Abb. 12.23 auf dem Bildschirm sehen.

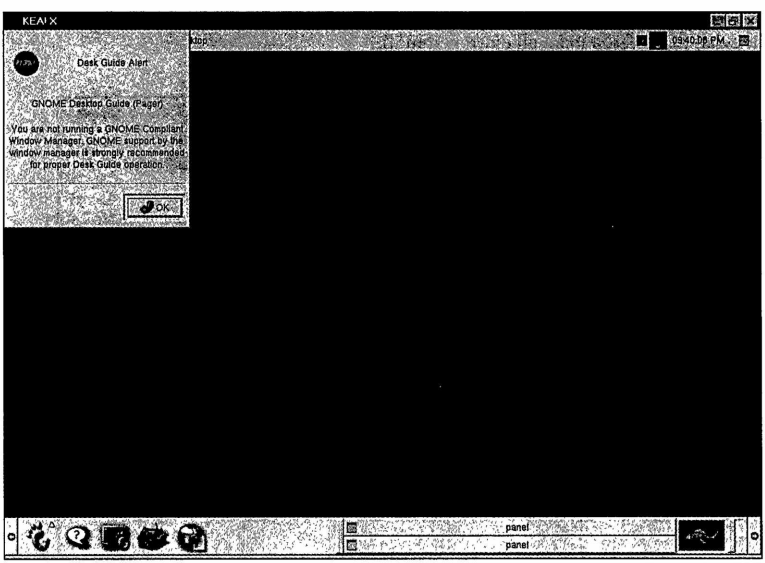


Abb. 12.23: Der Gnome-Desktop

Es wird die Fehlermeldung angezeigt, daß ein Fenstermanager verwendet wird, der mit Gnome nicht kompatibel ist. Da bisher kein anderer installiert ist, wird das in der Regel der *twm* sein, der natürlich nicht zu den neuesten gehört. Damit diese Fehlermeldung nicht wieder erscheint, richtet man sich einfach einen passenden ein. Das ist beispiels-

weise der Window Manager mit dem Namen *sawfish*. Passiert das nicht und sieht man nur einen leeren Bildschirm, hat man das System vermutlich als *root*-Benutzer gestartet. In diesem Fall beendet man den X-Server manuell per [Alt]+[Strg]+[Backspace] und versucht den Start noch einmal als »normaler« Benutzer. Es sollte jetzt ohne Probleme gelingen.

Ist *sawfish* installiert, muß man dem Gnome-Desktop nur noch beibringen, welchen Window Manager er verwenden soll. Das geschieht am einfachsten durch eine Umgebungsvariable in der Startdatei der Shell. Setzt man die in NetBSD standardmäßig verwendete die C-Shell ein, muß man in die Datei ».cshrc« im Heimatverzeichnis folgende Anweisung eintragen:

```
setenv WINDOW_MANAGER /usr/X11R6/bin/sawfish
```

Damit wird standardmäßig der Fenstermanager *sawfish* gestartet, der jetzt natürlich installiert sein muß, damit das funktionieren kann (sonst trägt man einfach einen anderen ein). Außerdem ist zu diesem Zeitpunkt empfehlenswert, eventuell vorhandene Verzeichnisse im Heimatverzeichnis, deren Namen mit ».gnome« beginnen, zu löschen. Diese und ihre Inhalte werden bei einem Neustart des Gnome-Desktops automatisch neu erzeugt, so daß jetzt auch der passende Fenstermanager darin aufgenommen wird, ohne daß man diesen manuell eintragen muß.

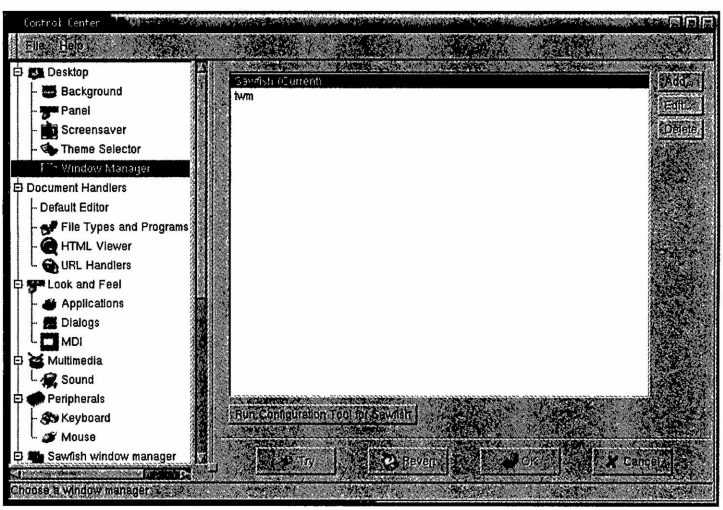


Abb. 12.24: Die Auswahl eines Fenstermanagers

Im laufenden Betrieb ist es ebenfalls möglich, den Fenstermanager zu ändern. Dafür bemüht man das Konfigurationswerkzeug von Gnome (der Werkzeugkasten in der unteren Leiste in Abb. 12.24). Wählt man darin den Punkt *Desktop/Window Manager*

aus, erscheint eine Liste mit den aktuell auf dem System installierten Fenstermanagern. Der aktuell laufende sollte in dieser Liste als *current* angezeigt werden, was aber leider nicht immer stimmt. Man sollte sich davon also nicht verwirren lassen.

Will man jetzt einen neuen auswählen, markiert man ihn in der Liste und klickt anschließend den Button *Try* an. Damit wird der entsprechende Fenstermanager gestartet, wenn alles klappt, und man sieht danach, wie die Fenster entsprechend neu aufgebaut werden. Ist der gewünschte der aktuelle, läßt er sich nicht über den Button *Try* starten. Ist dieser Fenstermanager tatsächlich aber nicht gestartet, muß man zuerst einen anderen auswählen und mit *Try* starten, bevor man danach den gewünschten endlich hochfahren kann. Da standardmäßig immer auch der *twm* mit installiert ist, gibt es so immer eine Alternative, die temporär benutzt werden kann.

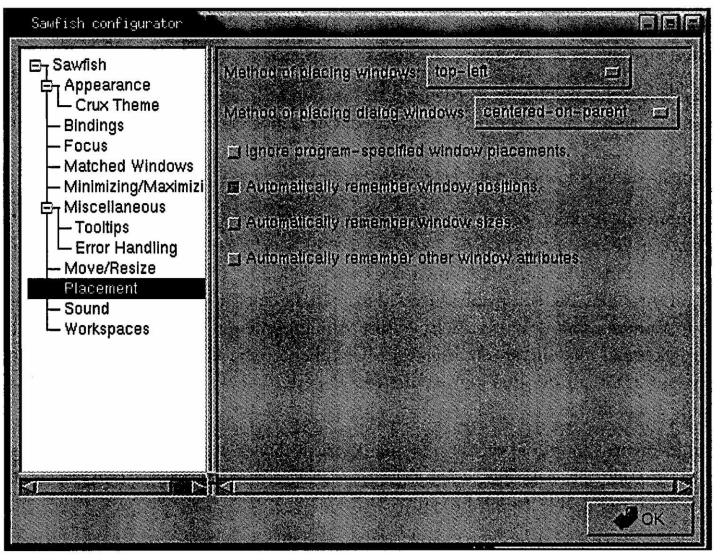


Abb. 12.25: Die Konfiguration des Fenstermanagers sawfish

War das Hochfahren des neuen Fenstermanagers erfolgreich, sieht man eine Dialogbox mit einer entsprechenden Meldung auf dem Bildschirm. Mit ihrer Hilfe speichert man die Einstellungen sofort, wenn man nur die Eingabetaste bestätigt beziehungsweise den Punkt *Save Session Now* auswählt.

Hat man den Fenstermanager *sawfish* installiert, gibt es für ihn noch einen extra Button, über den er speziell konfiguriert wird. Voraussetzung dafür ist, daß der *sawfish* der aktuelle Fenstermanager ist, weil erst dann ist der Button verfügbar ist. Außerdem muß er tatsächlich gestartet sein, da sonst die Konfiguration natürlich auch nicht funktionieren würde.

Es erscheint ein weiterer Dialog, in dem weitere Optionen eingestellt werden können. Dazu gehören beispielsweise das Aussehen der Fenster oder auch die Tastenkombinationen, mit deren Hilfe zwischen den einzelnen Fenstern umgeschaltet werden kann. Außerdem lassen sich noch Einstellungen in Bezug auf das Verschieben von Fenstern vornehmen oder man kann bestimmten Fensteraktionen (Verkleinern, Vergrößern und so weiter) Töne zuweisen.

12.11.3 Minimalsystem

Wie beim K-Desktop ist es auch bei Gnome möglich, nur ein Minimalsystem zu installieren, so daß man den eventuell knapp vorhandenen Plattenplatz für andere Programme nutzen kann. Beim Installieren des Systems muß man daran denken, den Platz für die »/usr«-Unterpartition zu vergrößern, denn auf dieser werden die meisten Daten gespeichert. Allerdings dürfen auch auf »/« nicht weniger als zirka 130 MByte zur Verfügung stehen, denn sonst lassen sich die Pakete nicht auspacken.

Das erste Paket, das installiert werden sollte, ist der Display-Manager *gdm*. Das gilt allerdings nur, wenn man das eine Megabyte, das er benötigt, zur Verfügung hat. Die Pakete, die dafür benötigt werden, sind aber auch für den Gnome-Desktop erforderlich, so daß man sie danach sowieso noch einrichten müßte.

Dazu gehören die Pakete *esound*, *gindent*, *glib*, *gnome-libs*, *gtk+*, *imlib*, *jpeg*, *libaudiofile*, *libglade*, *libungif*, *libxml*, *netpbm*, *ORBit*, *perl*, *png*, *pth* und *tiff*. Erst danach ist es möglich, das Paket *gdm* zu installieren. Die Namen der Pakete sind hier in alphabetischer Reihenfolge aufgeführt. Das bedeutet aber nicht, daß man sie auch tatsächlich in dieser Reihenfolge installieren kann. Sollten sich weitere Abhängigkeiten ergeben, ist es ratsam, erst diese Pakete zu installieren, um dann in die alphabetische Liste zurückzukehren. In dieser überspringt man die bereits installierten Pakete, so daß sich insgesamt nur die Reihenfolge etwas ändert.

Damit funktioniert schon mal der Displaymanager, doch der wird für den eigentlichen Desktop nicht benötigt. Allerdings läßt sich damit das System bequem starten, wie man im nächsten Kapitel nachlesen kann.

Das, was man für den Gnome-Desktop benötigt, ist das Basis-Paket *gnome-core*. Allerdings muß man zuvor wieder eine Reihe von Paketen installieren, von denen *gnome-core* abhängig ist. Dies sind in alphabetischer Reihenfolge *bonobo*, *control-center*, *db3*, *freetype-lib*, *gal*, *Gconf*, *gdk-pixbuf*, *gdk-pixbuf-gnome*, *ghostscript*, *ghostscript-fonts*, *gle*, *glibwww*, *glut*, *gnome-print*, *gnome-vfs*, *gtkhtml*, *ja-vflib-lib*, *libghttp*, *libiconv*, *libunicode*, *libwww*, *oaf*, *scrollkeeper*, *watanabe-vfont* und *xscreensaver-gnome*. Es handelt sich um eine recht große Reihe von Paketen, die vor *gnome-core* installiert werden müssen. Aber wenn das alles steht, kann man zumindest schon ein Minimalsystem starten.

Will man das System noch ein wenig feintunen, sollten noch die Applets mit dem Paket *gnome-applets* installiert werden. Allerdings wird dafür noch zusätzlicher Plattenplatz benötigt, in diesem Fall weitere zehn MByte.

Was jetzt noch fehlt, ist – wie im vorigen Kapitel auch – der passende Fenstermanager. Aus diesen Grund erscheint nach dem Start eine entsprechende Warnung.

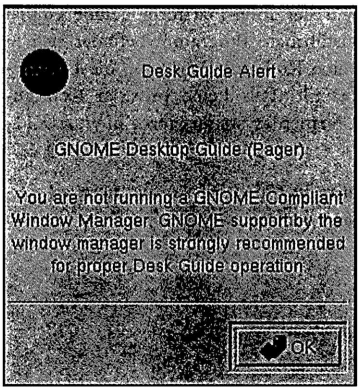


Abb. 12.26:
Es ist kein Gnome-kompatibler Fenstermanager installiert

In den Basispaketen von X Window ist ein Fenstermanager (*twm*) enthalten, der aber nicht kompatibel zu Gnome ist. Damit läßt sich der Desktop zwar bedienen, aber wirklich vernünftig ist so nicht zu arbeiten. Aus diesem Grund sollte ein anderer installiert werden, der aber Gnome-kompatibel sein muß. Da hier ja von einem Minimalsystem ausgegangen werden soll, scheiden die beiden großen, *AfterStep* und *sawfish*, im Prinzip aus. Damit bleibt nur der *IceWM* über, der zirka 780 Kbyte auf der Platte belegt und relativ klein ist.

Damit das System den Fenstermanager kennt, trägt man wieder die Umgebungsvariable *WINDOW_MANAGER* in die Startdatei der Shell ein. Welche Startdatei zum Einsatz kommt, hängt von der eingesetzten Umgebung ab, bei einer C-Shell ist dies die ».cshrc« im Heimatverzeichnis eines Benutzers. Steht der *IceWM* aber nicht zur Verfügung, muß man wohl oder übel den Fenstermanager *sawfish* benutzen.

Will man abschließend noch ein wenig Platz freimachen, sollte man die nicht benötigten Fenstermanager löschen. Damit werden einige Megabyte freigegeben, die besser für den *gdm* oder für *sawfish* genutzt werden sollten.

12.12 X für Puristen: Der twm

von Jörg Braun

Auf den bisherigen Seiten dieses Kapitel haben Sie einiges lesen, wie man die Oberflächen KDE und Gnome oder auch einen anderen Window Manager unter NetBSD zum Laufen bekommt. Auf einem schnellen Intel-Machine ist das auch ganz nett. Wer aber gern mit der Kommandozeile arbeitet und eigentlich nur X verwenden will um damit bequem mehrere Konsolen auf dem Bildschirm zu haben oder beispielsweise auf einem Mac68k eine deutsche Tastaturbelegung haben will muß nicht gerade ein mehrere hundert Mbyte großes Programmpaket installieren nur um eine Anwendung starten zu können. Sicher, es gibt auch etwas anspruchlosere Oberflächen wie KDE 3.0: Ein KDE 1.12, IceWM oder auch WindowMaker stellt viele Funktionen bei erheblich geringerem Ressourcenverbrauch zur Verfügung. Aber diese Pakete müssen auf der Festplatte installiert werden – der twm ist im Grundpaket von X bei NetBSD enthalten.

Der twm ist ein recht schlankes Programm, auf der Intel-Plattform gerade einmal 157130, bei m68k 125764 Byte groß. Das ausführbare Programm befindet sich in »/usr/X11R6/bin«, es benötigt aber auch noch eine Konfigurationsdatei, die Unix-typisch eine reine Textdatei ist und systemweit gilt (falls keine privaten Konfigurationsdateien angelegt werden) und unter dem Namen »system.twmrc« in »/usr/X11-R6/lib/X11/twm« abgelegt ist.

Die Beispieldatei in diesem Verzeichnis ist schon recht ordentlich konfiguriert, an einigen Stellen sollten trotzdem aber Änderungen vorgenommen werden.

```
#
# $Xorg: system.twmrc,v 1.3 2000/08/17 19:54:08 cpqbld Exp $
#
# Default twm configuration file; needs to be kept small to conserve string
# space in systems whose compilers don't handle medium sized strings.
#
# Sites should tailor this file, providing any extra title buttons, menus, etc.
# that may be appropriate for their environment.  For example, the most of the
# users were accustomed to uwm, the defaults could be set up not to decorate
# any windows and to use meta-keys.
#
# $XFree86: xc/programs/twm/system.twmrc,v 1.4 2001/01/17 23:45:08 dawes Exp $
#

NoGrabServer
RestartPreviousState
DecorateTransients
TitleFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
ResizeFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
MenuFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
IconFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
IconManagerFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
#ClientBorderWidth

Color
{
    BorderColor "slategrey"
    DefaultBackground "rgb:2/a/9"
    DefaultForeground "gray85"
    TitleBackground "rgb:2/a/9"
    TitleForeground "gray85"
    MenuBackground "rgb:2/a/9"
    MenuForeground "gray85"
    MenuTitleBackground "gray70"
    MenuTitleForeground "rgb:2/a/9"
    IconBackground "rgb:2/a/9"
```

```
    IconForeground "gray85"
    IconBorderColor "gray85"
    IconManagerBackground "rgb:2/a/9"
    IconManagerForeground "gray85"
}

#
# Define some useful functions for motion-based actions.
#
MoveDelta 3
Function "move-or-lower" { f.move f.deltastop f.lower }
Function "move-or-raise" { f.move f.deltastop f.raise }
Function "move-or-iconify" { f.move f.deltastop f.iconify }

#
# Set some useful bindings.  Sort of uwm-ish, sort of simple-button-ish
#
Button1 = : root : f.menu "defops"

Button1 = m : window|icon : f.function "move-or-lower"
Button2 = m : window|icon : f.iconify
Button3 = m : window|icon : f.function "move-or-raise"

Button1 = : title : f.function "move-or-raise"
Button2 = : title : f.raiselower

Button1 = : icon : f.function "move-or-iconify"
Button2 = : icon : f.iconify

Button1 = : iconmgr : f.iconify
Button2 = : iconmgr : f.iconify

#
# And a menus with the usual things
#
menu "defops"
{
"Twm"           f.title
"Iconify"       f.iconify
"Resize"        f.resize
"Move"          f.move
"Raise"         f.raise
"Lower"         f.lower
""              f.nop
"Focus"         f.focus
"Unfocus"       f.unfocus
"Show Iconmgr"  f.showiconmgr
```

```
"Hide Iconmgr"    f.hideiconmgr
""                f.nop
"Xterm"           f.exec "exec xterm &"
""                f.nop
"Kill"            f.destroy
"Delete"          f.delete
""                f.nop
"Restart"         f.restart
"Exit"            f.quit
}
```

Listing 12.1: Die »system.twmrc« in der Grundeinstellung

Diese doch recht übersichtliche Datei besteht aus vier Bereichen. Zuerst werden einige Einstellung für das Verhalten der Fenster definiert, im Block »Color« werden die Farben definiert, anschließend werden Funktionen für das Mausverhalten definiert und dabei der linken Maus das Menü »defops«, das anschließend definiert wird.

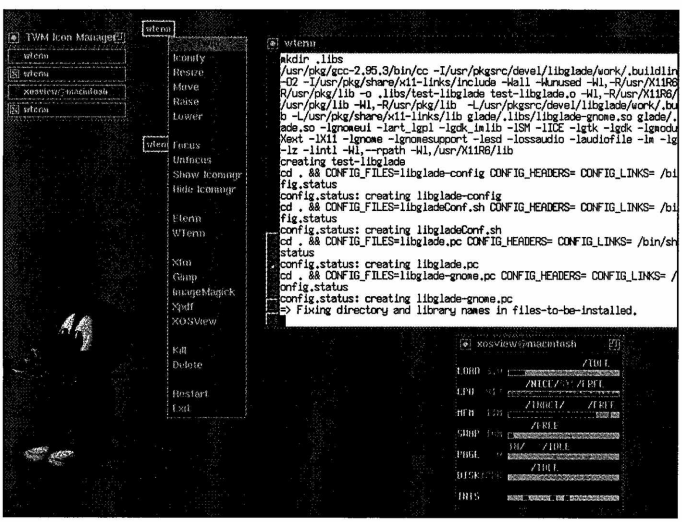


Abb. 12.27: Der twm, hier auf NetBSD/mac68k, erfüllt eigentlich alle Wünsche

Wie man das Verhalten des twm steuert erfährt man aus der recht umfangreichen Man-Pages die sich im Bereich 1 befindet. Sicher wird insbesondere störend empfunden, daß jedes Fenster manuell beim Start eines Programms mit der Maus plaziert werden muß. Man schaltet das ab, indem in den ersten Bereich der Datei die Zeichenkette *RandomPlacement* eingetragen wird. Daß der Inhalt von Fenstern beim Verschieben auf dem Desktop angezeigt wird regelt der Eintrag *OpaqueMove*.

Befehl	Funktion
ShowIconManager	Der Icon-Manager wird von vornherein angezeigt.
IconManagerGeometry	Definition der Position und Größe des Icon-Managers
SortIconManager	Einträge werden alphabetisch sortiert. Im Grundzustand werden Einträge immer angehängt.
NoCaseSensitive	Bei der Sortierung im Icon-Manager Groß- und Kleinschreibung ignorieren.
WarpCursor	Ein wiederhergestelltes Fenster erhält automatisch den Fokus.
RandomPlacement	Neues Fenster automatisch und zufällig plazieren, keine manuelle Festlegung mit der Maus.
OpaqueMove	Fensterinhalt wird beim Bewegen auf dem Desktop angezeigt nicht nur der Rahmen wie in der Voreinstellung.
NoTitle	Liste (durch Kommata getrennt) der Fenster die ohne Titelleiste angezeigt werden sollen.
NoDefaults	Keine Standard-Buttons in der Titelleiste der Fenster anzeigen.

Tabelle 12.9: Steuervariablen für das generelle Verhalten des twm

Es stellt auch kein Problem dar, das Menü selbst zu ändern. Auf einem Macintosh mit der berüchtigten Ein-Tasten-Maus ist es beispielsweise nicht möglich, in einem XTerm-Fenster zu scrollen. Man sollte deshalb Programme verwenden, bei denen das Scrollen besser geregelt ist. Auch *rxvt* und *aterm* lassen hier Komfort missen. Wunderbar funktionieren dagegen das Enlightenment-Terminal (»Eterm«) und das WindowMaker-Terminal (»wterm«).

Hierzu müssen nur die entsprechenden Einträge für das Menü geändert oder ergänzt werden, beispielsweise mit:

```
"Hide Iconmgr"  f.hideiconmgr
""              f.nop
"Eterm"         f.exec "exec Eterm &"
"WTerm "        f.exec "exec wterm &"
""              f.nop
"Xfm"           f.exec "exec xfm &"
"Gimp"          f.exec "exec gimp &"
"Xpdf"          f.exec "exec xpdf &"
"XOSView"       f.exec "exec xosview &"
""              f.nop
```

Wichtig ist, daß man hier dem Aufruf, der durchaus auch Parameter enthalten darf, das &-Zeichen nicht vergißt, da sonst die Oberfläche beim Aufruf des Menüpunkts blokkiert wird.

Das Ändern der Fensterdekorationen oder auch die Definition von Untermenüs ist ohne Probleme möglich.

Für zusätzliche Fensterdekorationen sind die folgenden Zeilen zuständig:

```
NoDefaults
LeftTitleButton ":menu" = f.menu "defops"
LeftTitleButton ":resize" = f.resize
RightTitleButton ":dot" = f.iconify
RightTitleButton "max" = f.zoom
RightTitleButton ":xlogo" = f.delete
```

Zu beachten ist, daß ohne den Eintrag »NoDefaults« die Originalflächen erhalten bleiben. Beim kompletten Neueinrichten darf diese Zeile also nicht fehlen.

Die Position der Fensterdekorationen geht aus den Beschreibungen klar hevor: »Left...« bedeutet, daß auf der linken Seite, »Right...«, daß auf der rechten Seiten ein Button zugefügt wird. Die Symbole werden in der Reihenfolge der Definition angelegt. Als Symbol wird bei der Angabe mit führendem Doppelpunkt eines der vier möglichen Standardsymbole verwendet. Zusätzliche Symbole müssen selbst gezeichnet werden. Dafür eignet sich vorzüglich das mit XFree86 mitgelieferte *bitmap*.

Das Programm *bitmap* dient dem Erzeugen von X-Pixmaps. Diese Grafikdateien sind reine Textdateien mit einer einfachen C-Syntax-artigen Beschreibung von Schwarzweißbildern. So sieht das in Abb. 12.28 dargestellte Bildchen im »Quelltext« beispielsweise folgendermaßen aus:

```
#define max_width 16
#define max_height 16
static unsigned char max_bits[] = {
   0xff, 0xff, 0xc1, 0x83, 0xe1, 0x87, 0xf1, 0x8f, 0xb9, 0x9d, 0xbd, 0xbd,
   0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0xbd, 0xbd, 0xb9, 0x9d,
   0xf1, 0x8f, 0xe1, 0x87, 0xc1, 0x83, 0xff, 0xff};
```

Listing 12.2: Der Quelltext von »max«

Wie in Listing 12.2 angegeben, müssen die Symbole 16x16 Pixel groß sein. Das ist auch die voreingestellte Größe in *bitmap*.

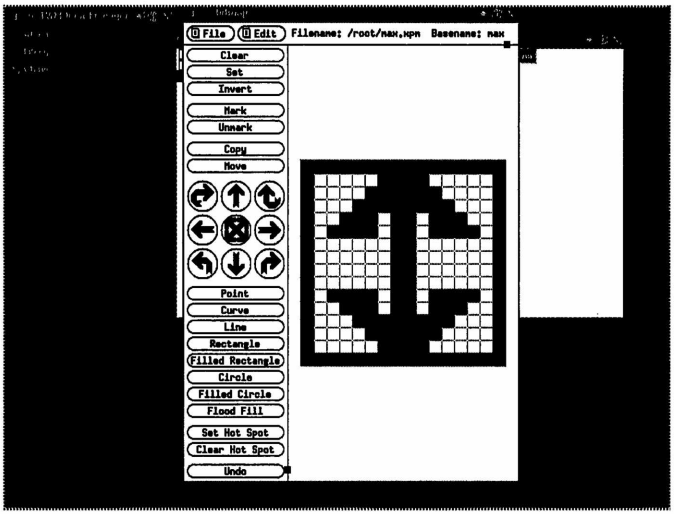


Abb. 12.28: Neue Fenstersymbole zeichnet man bequem mit dem Programm bitmap

Das oder die Bilder müssen anschließend entweder im Heimatverzeichnis des Benutzers oder im globalen Verzeichnis »/usr/X11R6/include/X11/bitmaps« abgelegt sein. Möchte man sie an anderer Stelle speichern, muß das in der Datei »/usr/X11R6/lib/X11/xdm/Xresources« oder einer lokalen Kopie ».Xresources« folgendermaßen bekanntgegeben werden:

```
*bitmapFilePath:/<Pfad_zu_den_Dateien>
```

Vergibt man für die Datei eine Endung (bei X-Pixmaps normalerweise ».xpm«) muß diese bei der Definition in twm-Beschreibungsdatei ebenfalls angegeben werden.

Die Änderungen wurden bisher immer global an der »system.twmrc« durchgeführt. Wer es komfortabler und an den User angepaßt haben will, legt dafür eine lokale Kopie ».twmrc« an. Wird diese Datei beim Start vom Window Manager gefunden, wird ausschließlich sie verwendet.

Bis jetzt wurde eigentlich nur etwas am Erscheinungsbild geändert. Der twm erlaubt aber auch das Strukturieren der Menüs. Dazu werden Untermenüs definiert. In sie wird anschließend vom Hauptmenü »Main«, definiert in »defops«, verzweigt:

```
menu "Xterm"                              # Name
{
"Terminals"     f.title                   # Name des Menüs (optional)
"Eterm"         f.exec "exec Eterm &"     # Funktionen
"WTerm"         f.exec "exec wterm &"
"XTerm"         f.exec "exec xterm &"
"Aterm"         f.exec "exec aterm &"
}
```

Es wird zuerst das Menü definiert und dann weiter unten in der Datei – Einträge müssen immer zuerst definiert werden und können dann später in der Datei verwendet werden – der entsprechende Eintrag im Hauptmenü festgelegt werden:

```
...
"XTerm"         f.menu "Xterm"
...
```

So verfährt man generell mit neuen Untermenüs. Auch Untermenüs von Menüs sind so kein Problem.

Der twm kennt die in Tabelle 12.10 aufgelisteten Funktionen.

Der eigenen Lust am Experimentieren und der Kreativität sind beim Einrichten des twm keine Grenzen gesetzt. Auf langsameren Maschinen ist sein Einsatz durchaus eine Überlgung wert. Es ist bei Bedarf ja auch jederzeit möglich, KDE- oder Gnome-Programm zu starten. Nur deren erster Start dauert wegen des notwendigen Ladens der benötigten KDE- und Gnome-Bibliotheken etwas länger.

Außer der Definition des Aussehens des Menüs und der Fensterdekorationen lassen sich auch Tasten (Hotkeys) und die Reaktion auf Maustasten neu einstellen. Die in Listing 12.1 im Bereich *useful bindings* angegebenen Werte sind auf manchen Rechnern, insbesondere solchen die keine Drei-Tasten-Maus besitzen, wenig hilfreich. Tabelle 12.11 listet die möglichen Tasten- und Mausfunktionen auf. Welche Belegungen überhaupt möglich ist, zeigt ein Aufruf des Programms *xmodmap* mit dem Parameter *-pm*:

```
[~] #xmodmap -pm
xmodmap:  up to 2 keys per modifier, (keycodes in parentheses):

shift       Shift_L (0x31),  Shift_R (0x3d)
lock        Shift_Lock (0x41)
control     Control_L (0x24),  Control_R (0x5f)
mod1        Alt_L (0x3f)
mod2
mod3        Mode_switch (0x61)
mod4
mod5
```

Hier dargestellt auf einem Mac68k mit deutscher X-Tastaturbelegung. Die Werte sind auf einer i386-Machine aber identisch. Man sieht daran, welche Belegungen überhaupt nutzbar sind und verwendet werden sollten.

Funktion	Beschreibung
f.menu "<Bezeichner>"	Menü *<Bezeichner>* anzeigen. Das Menü muß in der Datei bereits definiert sein.
f.function "<Bezeichner>"	Benutzerdefinierte Funktion *<Bezeichner>* ausführen. Die Funktion muß in der Datei bereits definiert sein.
f.iconify	Fenster iconisieren oder wieder vergrößern
f.resize	Fenstergröße ändern
f.zoom	Fenster maximieren oder Originalgröße wieder restaurieren.
f.move	Fensterposition auf dem Desktop ändern.
f.raiselower	Fenster auf dem Desktop nach vorn oder nach hinten verlagern.
f.focus	Fenster behält den Eingabefokus auch nach dem Verlassen des Mauscursors.
f.unfocus	Festen Fokus aufheben.
f.warpring "next"	Zum nächsten Fenster wechseln.
f.warpto "<Bezeichner>"	Zum Fenster *<Bezeichner>* wechseln.
f.showiconmgr	Icon-Manager anzeigen.
f.hideiconmgr	Icon-Manager verbergen.
f.delete	Fenster schließen.
f.destroy	Fenster gewaltsam beenden (*kill*).
f.identify	Fensterinformationen anzeigen.
f.restart	*twm* beenden und neu starten.
f.quit	*twm* beenden und Fensterdekorationen entfernen.
f.nop	*No-Operation-Funktion (macht nichts)*

Tabelle 12.10: twm-Funktionen

Schlüsselwort	Kürzel	Taste
shift	s	Shift
control	c	[Ctrl]
lock	l	[CapsLock]
meta	m	Normal [ESC] oder unbelegt
mod1	m1	Linke [Alt]-Taste
mod2	m2	[NumLock]
mod3	m3	Rechte [Alt]-Taste beziehungsweise [AltGr]
mod4	m4	Normalerweise unbelegt
mod5	m5	[Scroll]

Tabelle 12.11: Tastatur-Bindungen

Diese Tastenkombinaten lassen sich jetzt dazu nutzen, in Zusammenspiel mit der oder Maustasten zusätzliche Funktionen zu implementieren. Auf einem Macintosh mit Ein-Tastenmaus ist es beispielsweise nutzlos, Funktionen für die nicht vorhandene recht und mittle Taste definiert zu haben. Besser ist es, diese Funktionen auf Steuercodes in Zusammenarbeit mit den verfügbaren Tasten zu setzen:

```
Button1 =   : root : f.menu "defops"
Button1 = c : root : f.menu "Apps"
Button1 = s : root : f.menu "Xterm"

Button1 = s  : window|icon : f.function "move-or-lower"
Button1 = c  : window|icon : f.iconify
Button1 = m1 : window|icon : f.function "move-or-raise"

Button1 =   : title : f.function "move-or-raise"
Button1 = c : title : f.raiselower

Button1 =   : iconmgr : f.iconify
Button1 = c : iconmgr : f.iconify
```

Im einzelnen bedeutet daß, ein Tastendruck auf dem Desktop (»*root*«) das Hauptmenü, ein [Ctrl]+Tastendruck das Menü »Apps« und ein [Shift]+Tastendruck das Terminal-Menü zeigt. Die Einträge »: window« sind für Aktionen in normalen Fenstern, die »: title«-Einträge für Aktionen im Titelbalken und die » : iconmgr« im Icon-Manager zuständig.

Der twm läßt sich also hervorragend konfigurieren.

Es muß aber beachtet werden, daß der twm überhaupt nicht tolerant auf Fehler in der Konfigurationsdatei reagiert. Vergißt man beispielsweise einmal die Anführungszeichen, wird nicht einfach die fehlerhafte Zeile ignoriert, vielmehr wird anschließend eine riesige Liste von Fehlern produziert und der Window Manager steht. Das kann sich fatal auswirken. Hat man keine Textkonsole oder Remote-Login-Möglichkeit zur Verfügung, wo man X abzuschießen kann, bleibt nur noch ein Reset des Rechners.

Das FFS-Dateisystem ist zwar äußerst stabil und kaum aus dem Gleichgewicht zu bringen, trotzdem bleibt es auch unter NetBSD eine heikle Angelegenheit, wenn man die Maschine nicht ordnungsgemäß herunterfährt.

Zum Schluß noch ein Hinweis: Es ist recht krude, was so an Farbkombinationen auf Non-Intel-Geräten in Zusammenhang mit X angeboten wird. Offensichtlich wird bei XFree86 immer von der Farbpalette des i386 und seiner Grafikkarten ausgegangen. Man kann die Farben aber mit etwas Arbeit umdefinieren. Was angeboten wird und wie was heißt, erfährt man bequemsten mit dem Programm *xcolorsel*, das sich in den Package-Sources in der Rubrik »x11« befindet.

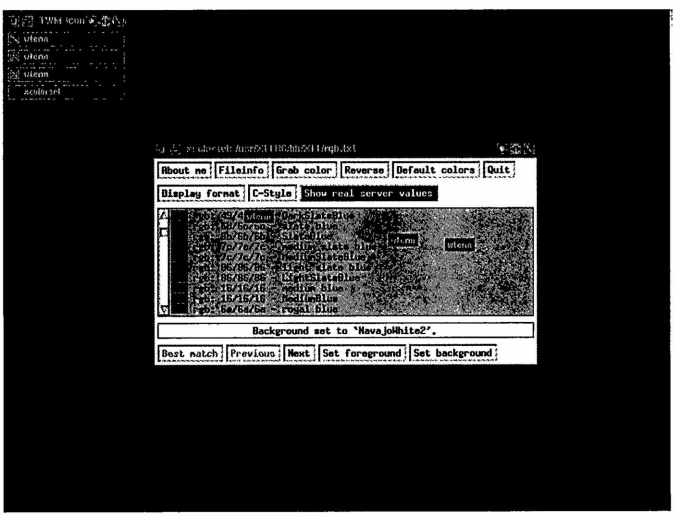


Abb. 12.29: Farbentschlüsselungen und Zuweisungen mit »xcolorsel«

12.13 X und die Tastatur

von Jörg Braun

In der Regel stimmt unter X ist die Tastatur nicht, zumindest wird sie sich in den seltensten Fällen genauso verhalten wie man es erwartet, sei es, daß die Backspace-Taste auf einmal nicht mehr so funktioniert, wie man es gewohnt ist oder daß man – im schlimmsten Fall – keinen deutschen Tastaturtreiber zur Verfügung hat.

Manchmal findet man zwar einen per Hand aufgebauten Tastaturtreiber, aber sicher kann man sich dessen nicht sein. Für solche Fälle gibt es Programme, mit denen man sich behilft. Mit dem X Eventtester läßt sich genau verfolgen, welche Taste welches Resultat und welchen Code zurückliefert. Das selbe gilt für Mausbewegungen und Maus-Tastendrücke.

Den gewünschten Code übergibt man dann *xmodmap* und hofft, daß es dann funktioniert.

Mit der Parameterkombinaten *-pke* bringt man xmodmap dazu, die aktuelle Tastaturbelegung auszugeben, leitet man dies in eine Datei um, hat man anschließend eine Konfigurationsdatei für das Keyboard. Man sollte sie global unter dem Namen ».Xmodmap« in »/usr/X11R6/lib/X11/xinit« ablegen. Eine in diesem Verzeichnis befindliche Datei wird beim Start mit xdm und kdm automatisch geladen. Beim Start über *startx* muß man die Zeile

```
xmodmap /usr/X11R6/lib/X11/xinit/.Xmodmap
```

in die Datei vor den Aufruf des Window Managers eintragen.

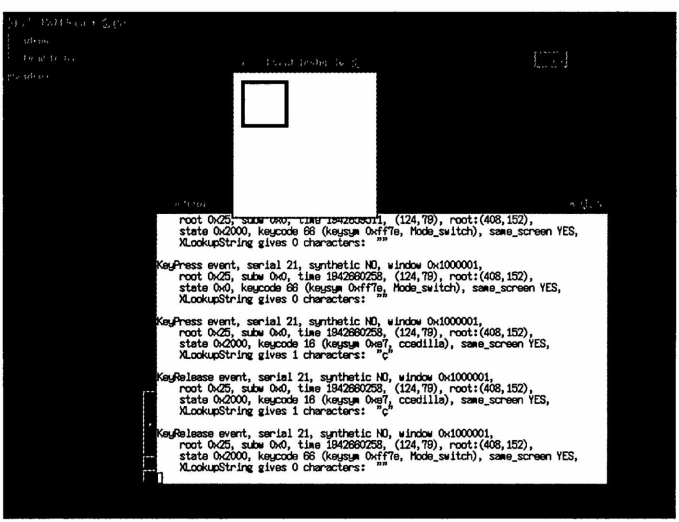


Abb. 12.30: Der Event-Monitor »xev«

Zu *xmodmap* gibt es auch ein grafisches Frontend. Es befindet sich den Package Sources unter »/usr/pkgsrc/x11/xkeycaps« und vereinfacht die Arbeit etwas. Das Programm erkennt die vorhandene Hardware und bietet dann schon recht zuverlässig die entsprechende Tastatur an. Falls man nicht zufrieden ist, kann man aber jederzeit eine andere Belegung wählen.

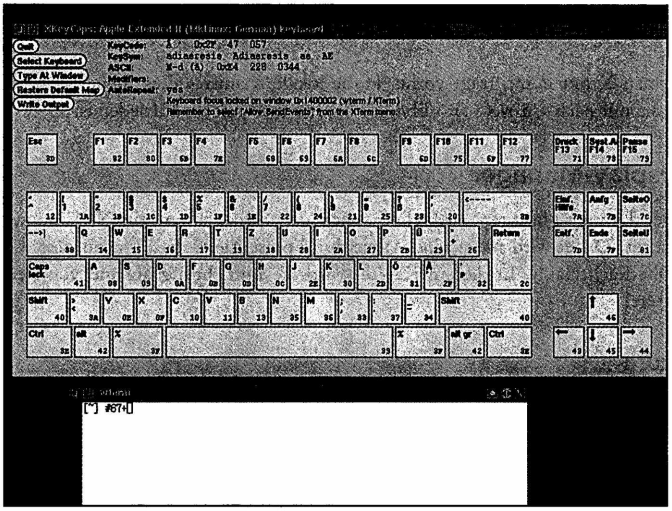


Abb. 12.31: Die Tastatur testen und ändern

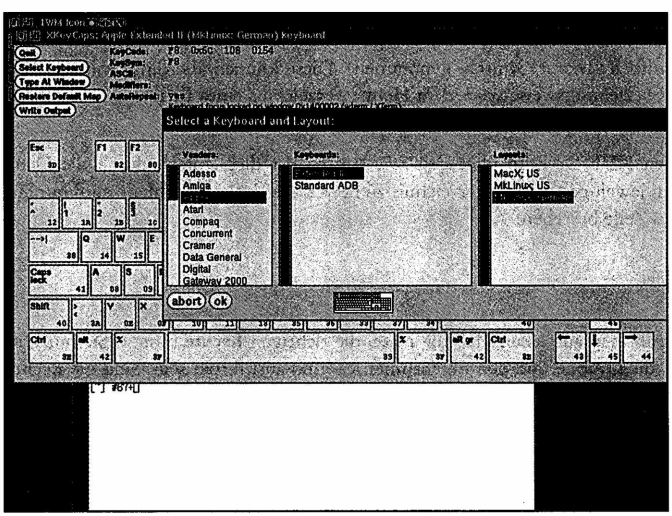


Abb. 12.32: Viel Auswahl bei der Tastatur hat man auf einem Macintosh nicht, aber es reicht. Auf dem Intel-PC ist die Auswahl vielfältiger.

Das Anzeigen allein führt zu nichts. Wenn man das Programm aber gleich richtig aufruft, hat man anschließend eine gute Konfigurationsdatei beisammen:

```
xkeycaps > ~/.Xmodmap &
```

Nun startet man X mit *startx*, trägt vorher in die lokale ».xinitrc« diese neue Datei an und schreibt sie nur nach »/usr/X11R6/lib/xinit«, wenn man mit ihr zufrieden ist.

12.14 Display-Manager

Mit Hilfe der Display-Manager wird gleich nach dem Hochfahren ein X-Window-System gestartet. Dabei wird meist eine Anmeldung präsentiert, die eben nur auf der grafischen Oberfläche läuft. In diesem Kapitel wird die Einrichtung anhand von Beispielen erläutert.

12.14.1 xdm

Will man beim Hochfahren des Rechners gleich das X-System starten, setzt das voraus, daß das X-Window-System korrekt konfiguriert ist. Danach fügt man in der Startdatei des NetBSD-Systems »/etc/rc.config« einfach die folgende Zeile ein:

```
xdm=yes
```

Das ist alles, nach einem Neustart steht der *xdm* zur Verfügung. Welche Programme danach aufgerufen werden, kann man in der Datei »/usr/X11R6/lib/X11/xdm/Xsession« festlegen (bei einem Start über das Skript *startx* wird standardmäßig die Datei »/usr/X11R6/lib/X11/xinit/ xinitrc« genommen). Diese können allerdings mit den Dateien *.xsession* beziehungsweise *.xinitrc* im Heimatverzeichnis eines Benutzers überschrieben werden, sie haben Vorrang vor den globalen Dateien.

Will man den KDE-Desktop über den *xdm* starten, ist dies einfach durch die Datei ».xsession« im jeweiligen Heimatverzeichnis zu bewerkstelligen. Der Inhalt muß nur aus der folgenden Zeile bestehen:

```
exec startkde
```

Allerdings muß man aufpassen, daß die Datei die richtigen Rechte besitzt. Sie muß für alle Besitzer, die Gruppe und den Eigentümer les- und ausführbar sein. Sind die Rechte nicht vorhanden, setzt man sie mit dem folgenden Befehl:

```
chmod 555 .xsession
```

Außerdem ist es erforderlich, daß die Datei dem jeweiligen Benutzer und der zugehörigen Gruppe gehört. Das kann man gegebenenfalls mit den Befehlen *chown* (Benutzer) und *chgrp* (Gruppe) ändern.

Auf diese Weise kann man auch andere Fenstermanger beziehungsweise Desktops starten. Dazu paßt man die Datei *.xsession* entsprechend an. Will man also beispielsweise den Fenstermanager *IceWm* starten, trägt man folgendes in die Datei ein:

```
exec /usr/X11R6/bin/icewm
```

wobei gegebenenfalls auch ein

```
exec icewm
```

funktionieren sollte. Für den Start des Gnome-Desktops muß die Datei folgenden Inhalt haben:

```
exec /usr/X11R6/etc/gdm/gnomerc
```

beziehungsweise

```
exec /usr/X11R6/etc/gnomerc
```

Zu beachten ist in diesem Zusammenhang, daß der für Gnome auszuführende Fenstermanager wieder über die Umgebungsvariable *WINDOW_MANAGER* gesetzt werden sollte. Wenn der Start des Fenstermanagers beziehungsweise des Desktops nicht gelingt, sind die Zugriffsrechte und der Besitzer der Datei ».xsession« zu überprüfen. Gegebenenfalls muß man diese, wie zuvor beschrieben, ändern.

XDMCP mit xdm

Ist der X-Displaymanager *xdm* beim Hochfahren aktiviert, besteht auch die Möglichkeit, den Desktop über das Netzwerk per XDMCP weiterzuleiten. Dazu wird aber ein entsprechender X-Server auf dem Client-Rechner (mit geeigneter Software funktioniert das sogar auf einem Windows-Rechner) benötigt. Standardmäßig ist diese Option aber ausgeschaltet, so daß sie erst aktiviert werden muß.

Dazu muß man die Konfigurationsdatei »xdm-config« im Verzeichnis »/usr/X11-R6/lib/X11/xdm« entsprechend anpassen. Ziemlich am Ende der Datei befindet sich die Zeile

```
DisplayManager.requestPort:  0
```

Aufgrund dieses Eintrags wird nicht auf entsprechende Anfragen gewartet, die über das Netzwerk kommen. Will man dennoch den XDMCP-Dienst zulassen, kommentiert man die Zeile einfach aus. Zu beachten ist dabei aber, das dafür kein Doppelkreuz, sondern ein Ausrufezeichen an den Anfang der Zeile gesetzt werden muß!

```
!DisplayManager.requestPort:  0
```

Die Änderungen werden erst wirksam, nachdem der Display-Manager neu gestartet wurde. Am unkompliziertesten ist der Neustart des Rechners. Alternativ wäre auch das Beenden des xdm-Prozesses möglich, allerdings scheint nach dem manuellen Abbruch

der Neustart des *xdm* nicht automatisch zu funktionieren. Aus diesem Grund ist es einfacher, den Rechner neu zu starten.

Sollte das nicht funktionieren, muß man noch an einer weiteren Datei Änderungen vornehmen. In der Datei »/usr/X11R6/lib/X11/Xaccess« findet man zwei Zeilen, die jeweils mit der Kombination

```
#*
```

beginnen. Wenn man jetzt in beiden Zeilen jeweils das Doppelkreuz entfernt und den Rechner danach neu startet, sollte er im Netzwerk zu sehen sein und gegebenenfalls den entsprechenden XDM-Bildschirm präsentieren.

12.14.2 gdm

Paket	Beschreibung
esound	Enlightened Sound Daemon.
gindent	GNU-Formatierungswerkzeug für C-Dateien.
glib	Nützliche Routinen für die C-Programmierung.
gnome-libs	Gnome-Bibliotheken.
gtk+	Werkzeug zum Erzeugen von X11-Benutzerschnittstellen.
imlib	Bildbearbeitungsbibliothek für X11.
jpeg	JPEG-Kompressionswerkzeuge.
libaudiofile	Soundbibliothek für SGI-Audiodateien
libglade	Laufzeitinterpreter für GLADE-GUI-Dateien.
libungif	Bibliothek zur Bearbeitung von GIF-Dateien.
libxml	Parser für XML-Dateien.
netpbm	Werkzeug zur Konvertierung von Grafikdateien.
ORBit	CORBA ORB für C.
perl	Die Skriptsprache Perl.
png	Bibliothek zur Bearbeitung von PNG-Dateien.
pth	Portable GNU Thread-Bibliothek.
tiff	Bibliothek zum Bearbeiten von TIFF-Dateien.

Tabelle 12.12: Die vom gdm unter NetBSD 1.6 benötigten Pakete

Einrichtung

Der Gnome-Display-Manager *gdm* wird als separates Paket bei der NetBSD-Distribution mitgeliefert, so daß er auch eingesetzt werden kann, wenn man einen anderen als den Gnome-Desktop bevorzugt. Das zugehörige Package befindet sich in »/usr/pkgsrc/x11/gdm«. Allerdings benötigt es noch diverse Bibliotheken beziehungsweise Pakete, die man gegebenenfalls nachinstallieren muß. Welche das sind, ist in Tabelle 12.12 nachzulesen.

Nach der Installation muß man dann nur noch das System so einstellen, daß es den *gdm* automatisch beim Hochfahren statt des XDM startet. Dazu ist es erst einmal erforderlich, den Eintrag

```
xdm=yes
```

in der Datei »/etc/rc.conf« auszukommentieren (mit einem Doppelkreuz), sofern er denn vorhanden ist. Anschließend stellt sich die Frage, wie man den *gdm* startet. Der erste Weg wäre wahrscheinlich, den Aufruf in die Datei »/etc/rc« aufzunehmen. Das funktioniert, hat aber mitunter den Nachteil, daß jetzt keine Konsolen mehr verfügbar sind. Das wäre vermutlich nicht weiter schlimm, aber es geht auch ein wenig eleganter.

Dazu nutzt man die Analogie zwischen den Befehlen in der Datei »/etc/rc.conf« und den Dateien im Verzeichnis »/etc/rc.d«. Sie tragen dann in die Datei »/etc/rc.conf« nur die Zeile

```
gdm=yes
```

ein und schon wird der *gdm* beim Hochfahren gestartet. Außerdem stehen die Terminals (zumindest einige) weiter zur Verfügung. Doch bevor das klappt, muß man noch ein wenig Hand anlegen. Man kopiert dazu die Datei »/etc/rc.d/xdm« einfach nach »/etc/rc.d/gdm« um. Dadurch ist zudem sichergestellt, daß der *gdm* ungefähr an der Stelle im Init-Prozeß gestartet wird wie der *xdm* auch (sofern man die Befehle in der *REQUIRE*-Zeile nicht ändert). Außerdem bleiben die Rechte erhalten (les- und ausführbar für alle Benutzer, 555). Würde man jetzt den Rechner neu starten, würde man weiterhin den *xdm* sehen, so daß man auch noch den Inhalt der Datei »/etc/rc.d/gdm« anpassen muß. Zu diesem Zweck ersetzt man einfach den String *xdm* durch ein *gdm*. Der Inhalt sollte dann folgendermaßen aussehen:

```
# PROVIDE: gdm
# REQUIRE: DAEMON LOGIN wscons
# KEYWORD: shutdown

. /etc/rc.subr

name="gdm"
rcvar=$name
command="/usr/X11R6/bin/${name}"
```

```
pidfile="/var/run/${name}.pid"
required_files="/usr/X11R6/etc/gdm/gdm.conf"
extra_commands="reload"

load_rc_config $name
run_rc_command "$1"
```

Anschließend wird die Datei abgespeichert und der Rechner neu gestartet. Nun sieht man statt des X-Display-Managers den Gnome Display-Manager auf dem Desktop. In der aktuellen Version des gdm-Pakets muß man sich übrigens nicht mehr die Mühe machen, die Datei in »/etc/rc.d« manuell zu erzeugen. Stattdessen reicht es aus, die Datei mit dem Namen »gdm« aus dem Verzeichnis »/usr/X11R6/etc/rc.d« in das Verzeichnis »/etc/rc.d« zu kopieren. Anschließend muß man nur den Eintrag

```
gdm=YES
```

in die Datei »/etc/rc.conf« aufnehmen und schon ist man fertig.

Wenn Gnome installiert ist, sollte sich dieser eigentlich als Default-System aus dem *gdm* heraus starten lassen (klappte aber in früheren Versionen nicht, weil Rechte falsch gesetzt waren), ein *twm* fährt nach der Auswahl von *Xsession* hoch. Untersucht man das Ganze jetzt etwas genauer, findet man den Desktop auf Konsole 2 statt auf 5, die anderen drei (1, 3 und 4) lassen sich aber ohne Probleme nutzen. Insgesamt scheint das System ein wenig instabil zu sein, was aber auch kein Wunder ist, denn Konsole 2 ist jetzt doppelt belegt. Außerdem sieht man auf Konsole 1 eine Reihe von Fehlermeldungen, die mit dem *gdm* zusammenhängen.

Aus diesem Grund ist jetzt ein wenig Nacharbeit erforderlich, um das System wieder stabil zu bekommen. Sollte sich das Gnome-System nicht starten lassen, sind erst einmal die Rechte der Datei *Gnome* im Verzeichnis »/usr/X11R6/etc/gdm/Sessions« auf les- und ausführbar für alle zu ändern, um zumindest diese Fehlermeldung zu beseitigen. Das funktioniert am einfachsten mit der folgenden Sequenz, für die man aber als root-Benutzer angemeldet sein sollte (alternativ erhält man die erforderlichen Rechte auch mit dem Kommando *su*).

```
cd /usr/X11R6/etc/gdm/Sessions
chmod 555 Gnome
cd
```

Startet man jetzt den Rechner neu, sieht man zumindest die zugehörige Fehlermeldung nicht mehr, alles andere ist aber gleichgeblieben.

Die nächste unter Umständen zu ändernde Einstellung betrifft die Datei mit der PID des gdm-Prozesses. Betrachtet man die Konfigurationsdatei »gdm.conf« im Verzeichnis »/usr/X11R6/etc/gdm« näher, findet man dort bei der Option *PidFile* die Angabe */var/run/xdm.pid*. Dies bedeutet, daß der gdm-Prozeß seine PID in eine Datei mit dem Namen »xdm.pid« schreibt (und nicht »gdm.pid«!). Aus diesem Grund sollte man jetzt

die Datei »/etc/rc.d/gdm« ändern, damit auch hier die korrekte Datei angegeben wird. Die entsprechende Zeile lautet dann wie folgt:

```
pidfile="/var/run/xdm.pid"
```

Alternativ ist aber auch nur die Änderung der gdm-Konfigurationsdatei möglich. In der Zeile *PidFile* sollte dann »/var/run/gdm.pid« stehen, was in der neuen Version automatisch der Fall ist. Damit ist ein weiteres Problem gelöst. Bleibt nur noch die Sache mit dem korrekten Terminal übrig.

Ohne weitere Angaben startet der *gdm* immer noch auf dem zweiten Terminal, was zu Überschneidungen mit der hier eingerichteten Konsole führt (man erreicht diese mit der Tastenkombination [Strg]+[Alt]+[F2]). Nun kann man natürlich daran gehen, die Konsole in der Datei »/etc/ttys« abzuschalten. Die entsprechende Zeile würde dann lauten:

```
ttyE1  "/usr/libexec/getty Pc"         vt220        off       secure
```

Doch irgendwie ist auch das nicht der Weisheit letzter Schluß, denn der *xdm* hat das X-Window-System auf Konsole 5 gestartet. Hierzu wurde übrigens die Datei *Xservers* im Verzeichnis */usr/X11R6/lib/X11/xdm* verwendet, in der nach der Angabe des Servers auch das entsprechende Terminal spezifiziert wird. Nur leider scheint diese vom *gdm* nicht ausgewertet zu werden, denn sonst würde es zu diesen Konflikten nicht kommen.

Schaut man sich den Inhalt der Datei »gdm.conf« im Verzeichnis »/usr/X11R6/etc/gdm« noch einmal an, findet man zum Abschluß den Abschnitt *{servers}*. In diesem ist eingetragen, welche Server gestartet werden. Zusätzlich steht dort ein Hinweis, daß man explizit eine Konsole angeben kann. Nur leider ist dieser für Linux gedacht, so daß erst mal wieder ein wenig Frust aufkommt.

Nimmt man den Hinweis aber ernst und probiert etwas, wird man feststellen, daß das Ganze auch unter NetBSD funktioniert. Damit sollte die Zeile wie folgt aussehen:

```
0=/usr/X11R6/bin/X vt5
```

Fährt man nach dem Abspeichern den Rechner erneut hoch, erscheint zwar immer noch eine Fehlermeldung des gdm, aber das X-Window-System wird jetzt auf Konsole 5 gestartet. Konsole 2 ist wieder verfügbar und das ganze NetBSD-System läuft insgesamt stabil.

Konfiguration

Die eigentliche Konfiguration des gdm findet in der schon erwähnten Datei »gdm.conf« im Verzeichnis »/usr/X11R6/etc/gdm« statt. Allerdings scheint diese nicht immer so zu funktionieren, wie man es erwarten könnte. Nach dem ersten Hochfahren des *gdm* werden alle Meldungen in Englisch angezeigt. Ein Ändern der Option *DefaultLocale* im Abschnitt *{greeter}* auf *german* sollte eigentlich die Meldungen in Deutsch herbeizaubern,

tatsächlich bleiben sie aber so, wie sie sind. Sie wechseln erst die Sprache, nachdem man sich an- und abgemeldet hat und man vorher im gdm-Menü *Sprache* den Punkt *A-M/German* ausgewählt hat. Nach dem Abmelden in der X-Sitzung scheinen die lokalen Dateien des gerade angemeldeten Benutzers gelesen zu werden, so daß eine Umstellung der Sprache erfolgt. Ansonsten wird der Inhalt der Konfigurationsdatei aber auch beim ersten Hochfahren des *gdm* gelesen, was man leicht selbst überprüfen kann, indem man eine Option ändert und den Rechner danach neu startet.

Doch im einzelnen: Die Konfigurationsdatei »gdm.conf« selbst ist, wie bei Unix-Systemen üblich, eine reine Textdatei. Sie ist in Abschnitte aufgeteilt, die durch Überschriften in eckigen Klammern beschrieben werden. Danach folgen dann die einzelnen Optionen, wobei die Form

```
Option = Wert
```

eingehalten wird. Für den Begriff *Wert* kann dabei eine Ziffer oder auch ein String beziehungsweise ein Verzeichnisname stehen. Bei Optionen mit nur zwei Auswahlmöglichkeiten steht eine 0 für deaktiviert und eine 1 für aktiviert. Alternativ können auch die booleschen Werte *false* (0) und *true* (1) genommen werden.

Der Abschnitt {daemon}

In diesem Abschnitt befinden sich die Optionen zur Konfiguration des gdm-Dämons. Damit wird prinzipiell das Aussehen vom *gdm* festgelegt. Bei allen Optionen werden zusätzlich die Default-Werte mit angegeben.

Der Abschnitt {security}

In diesem Abschnitt befinden sich Optionen, die die Sicherheit betreffen. Bei einigen Systemen kann man sich beispielsweise nicht als root-Benutzer über das Netzwerk anmelden. Ob das bei Ihrem Rechner so bleiben muß, bleibt letztendlich Ihnen selbst überlassen.

Der Abschnitt {xdmcp}

Diese Sektion enthält alle Optionen, die mit dem Einloggen über das Netzwerk über das XDMCP-Protokoll zu tun haben. Zu beachten ist in diesem Zusammenhang, daß die Reaktion beim *gdm* auf entsprechende Anfragen in der Regel abgeschaltet ist. Man muß also nicht nur die Zugriffsdateien des Systems anpassen, sondern auch die Konfigurationsdatei des *gdm*, damit das Einloggen möglich ist.

Der Abschnitt {gui}

Die Optionen in dieser Sektion dienen nur dazu, ein Thema und die Größe von Icons zu setzen. Meist kann man sie aber so lassen, wie sie sind.

Der Abschnitt {greeter}

Auch beim *gdm* ist es möglich, sich eine Liste mit den anzuzeigenden Benutzern ausgeben zu lassen. Allerdings hängt es vom eingesetzten System ab, ob dieser als *Greeter* bezeichnete Bereich standardmäßig dargestellt wird oder nicht. Einzelne Benutzer werden durch Icons repräsentiert, wobei ein Standard-Icon gesetzt ist. Zusätzlich kann man auch Icons nehmen, die ein Benutzer selbst für sich in seinem Heimatverzeichnis unter dem Namen »$HOME/.gnome/photo« angelegt hat. Wird ein solches Icon gefunden, wird es in der entsprechenden Liste angezeigt. Zu beachten ist dabei, daß der Name *photo* sein muß und daß dieser keine Dateinamenserweiterung enthält. In der Regel ist man auf der sicheren Seite, wenn als Format für die Icon-Datei das Grafikformat PNG gewählt wird.

Außerdem ist es in dieser Sektion möglich, den Begrüßungsstring zu ändern oder das System-Menü, das die Punkte zum Herunterfahren und Neustarten enthält, zu deaktivieren.

Der Abschnitt {chooser}

Ist der XDMCP-Chooser eingeschaltet, kann man hier angeben, welche Icons beispielsweise für die einzelnen Rechner verwendet werden sollen und wo sie zu finden sind.

Der Abschnitt {debug}

In diesem Abschnitt kann man Debugging-Features einschalten. Es gibt nur die Option *Enable*, die aber normalerweise auf *false* gesetzt, also deaktiviert ist. Man sollte es, wenn möglich, auch dabei belassen.

Der Abschnitt {servers}

Diese Sektion ist ein Kontrollabschnitt für den X-Server. Jede Zeile beginnt mit der lokalen Display-Nummer und endet mit dem Kommando, das erforderlich ist, um den Server zu starten. Zusätzlich fügt der *gdm* bei einem Aufruf die Optionen

```
-auth <ServAuthDir>/:n.Xauth :n
```

an, wobei *ServAuthDir* durch das in der Konfigurationsdatei angegebene Verzeichnis ersetzt wird. Der Buchstabe *n* bezeichnet schließlich noch die lokale Display-Nummer.

Die Standardeinstellung lautet:

```
0=/usr/X11R6/bin/X
#1=/usr/X11R6/bin/X
```

Zusätzlich zur Konfigurationsdatei *gdm.conf* gibt es noch eine weitere Datei, die sich nach dem ersten Anmeldevorgang über den *gdm* im Heimatverzeichnis befindet. Sie besitzt den Namen *gdm* und ist im Verzeichnis »$HOME/.gnome/« zu finden.

Es handelt sich wieder um eine reine Textdatei, die in der Sektion *{session}* die Daten der letzten Sitzung enthält. Dabei werden in der Regel die beiden nachfolgenden Optionen verwendet.

Option	Beschreibung
lang	Damit wird die Sprache gespeichert, die bei der letzten Sitzung eingestellt wurde.
last	Welches System als letztes gestartet wurde, kann man bei dieser Option ablesen. Die entsprechende Oberfläche wird wieder aufgerufen, wenn man im *gdm* den Menüpunkt *Sitzung/Letzte* auswählt.

Tabelle 12.13: Die Optionen der lokalen Konfigurationsdatei

Unterverzeichnisse

Im Verzeichnis, in dem die Konfigurationsdateien gespeichert sind, gibt es noch weitere Unterverzeichnisse, in denen Dateien für spezielle Funktionen enthalten sind. Eine Beschreibung befindet sich in der nachfolgenden Tabelle. Zu beachten ist, daß die Standardnamen der Pfade über die Optionen *DisplayInitDir*, *PostSessionScriptDir*, *PreSessionScriptDir* und *SessionDir* in der Konfigurationsdatei »gdm.conf« überschrieben werden können.

In der Regel befinden sich in den einzelnen Verzeichnissen nur die Default-Skripte, bis auf das Sessions-Verzeichnis. Dort sind die Skripte zum Starten der Oberflächen enthalten. Damit funktioniert die Anmeldung in der Regel ohne Probleme, so daß man im Prinzip keine weiteren mehr hinzufügen muß. Allerdings sollte man sich überlegen, ob man das Default-Skript im Verzeichnis *Sessions* den eigenen Gegebenheiten anpassen sollte.

Startmöglichkeiten

Die Sitzungen, die im gdm-Menü *Session* (*Sitzung*) angezeigt werden, entsprechen den Namen von Dateien im Verzeichnis »/usr/X11R6/etc/gdm/Sessions« (den Ordner kennen Sie bereits). Bei den Dateien handelt es sich meist um Skripte, mit deren Hilfe die entsprechenden Desktops aufgerufen werden können. Aber man kann natürlich bei Bedarf weitere einfügen, wenn das gewünscht ist. Man muß nur daran denken, daß die Dateien in diesem Verzeichnis les- und ausführbar für alle Benutzer (555) sein müssen.

Alternativ kann man auch das mitgelieferte Programm *gdmconfig* zur Konfiguration der Optionen nutzen. Es läßt sich beispielsweise in einem X-Terminal starten.

Allerdings treten beim Start eine Reihe von Fehlern auf, so daß das Programm nicht richtig zu bedienen ist. Aus diesem Grund sind auch nur die Optionen der Hauptgruppe *Basic* verfügbar, also die Seiten *Login appearance*, *Login behaviour* und *Face browser*. Leider lassen sich eventuell veränderte Optionen auch nicht übernehmen, so daß das Programm eigentlich nur zur Anzeige der Bildchen gebraucht werden kann.

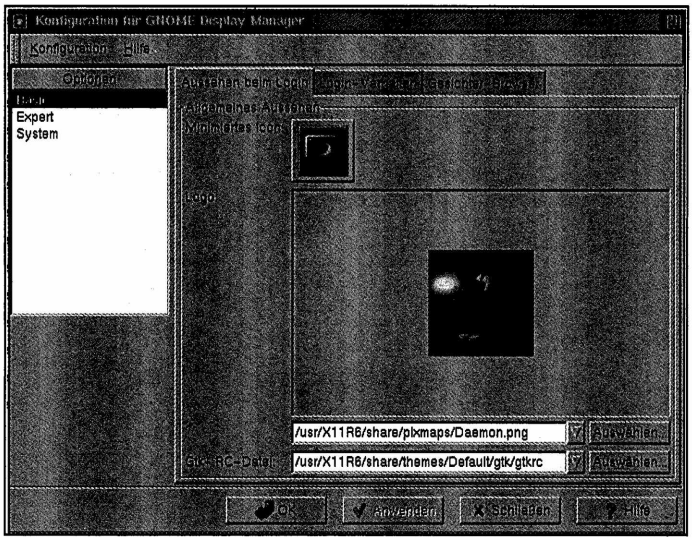


Abb. 12.33: Die grafische Konfiguration der gdm-Optionen

XDMCP

Will man sich auch über das Netzwerk per XDMCP-Protokoll auf den NetBSD-Rechner anmelden können, muß man ebenfalls die Konfigurationsdatei »/usr/X11-R6/etc/gdm/gdm.conf« anpassen. Die erforderliche Option befindet sich im Abschnitt *{xdmcp}*. Setzt man *Enable* auf *true*, kann man sich nach einem Neustart auch von einem entfernten Rechner her anmelden.

Zu beachten ist in diesem Zusammenhang, daß nun nicht mehr alle Menüpunkte des *gdm* zur Verfügung stehen. Auf dem entfernten Host wird das System-Menü ausgeblendet, mit dessen Hilfe man den Rechner herunterfahren oder neu starten kann.

Außerdem werden wieder alle Menüpunkte und Meldungen in Englisch angezeigt, unabhängig von der Einstellung in der Konfigurationsdatei. Das ändert sich auch nicht durch An- und Abmelden, da nun jedesmal eine neue (erste) Verbindung aufgebaut wird, so daß die Daten vorheriger Sitzungen nicht verfügbar sind.

Bei meinen Tests mit einem KEA! X-Server auf einem Windows-Rechner schlug mitunter die Übertragung des *gdm* schon im Anfangsstadium fehl, nach einiger Zeit trat ein Timeout auf. Bewährt hat sich aber die Methode, die Wahl des Rechners im entsprechenden Dialog des KEA! X-Servers erst nach einiger Zeit auszuführen, das heißt, die Dialogbox eine Zeitlang zu ignorieren. Danach klappte meist die Anmeldung, wobei es aber schneller ging, als in der Konfigurationsdatei die spezielle Konsole (vt5) nicht angegeben wurde.

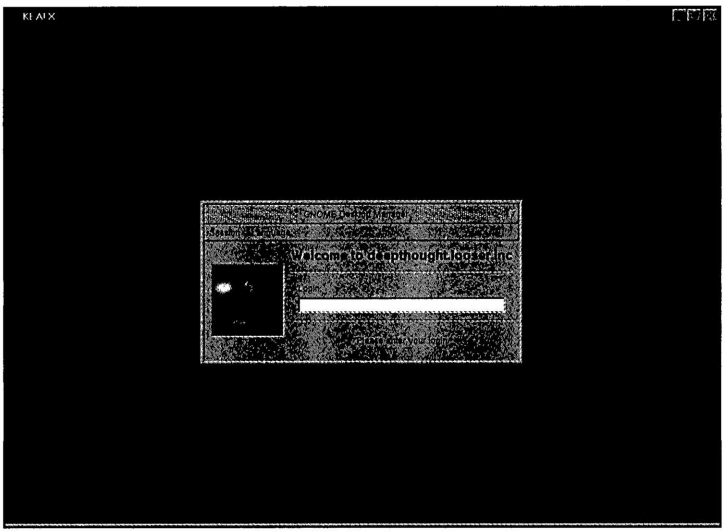


Abb. 12.34: Der gdm auf einem entfernten Rechner, hier eine Windows-Maschine mit dem X-Server KEA

Ob das bei Ihrem X-Server auf einem entfernten Rechner auch so ist, müssen Sie bei Ihrem System selbst ausprobieren. Auf jeden Fall muß lokal die fünfte Konsole (oder die sechste...) angegeben werden, da es sonst zu Instabilitäten auf dem NetBSD-Rechner kommt.

Ausschalten

Wenn man doch lieber einen anderen Display-Manager oder aber keinen einsetzen will, kann man die entsprechende Zeile *gdm=YES* in der Konfigurationsdatei »/etc/rc.conf« einfach auskommentieren. Bei einem Neustart wird der gdm nicht mehr automatisch gestartet, aber man sieht stattdessen eine Fehlermeldung. Diese kann man beseitigen, indem man die Auskommentierung aufhebt und das *YES* in *NO* verändern, oder man einfach die Datei »/etc/rc.d/gdm« löscht. Diese würde dann für einen erneuten Einsatz nicht mehr zur Verfügung stehen, so daß die erste Möglichkeit sicherlich die bessere Lösung ist. Ein Umbenennen würde übrigens auch nicht weiterhelfen, da die entsprechende Fehlermeldung dann den Namen der umbenannten Datei enthalten würde.

12.14.3 kdm

Auch den Display-Manager des KDE-Pakets, der auf den Namen *kdm* gehorcht, muß man von Hand einrichten, sofern man ihn zum Anmelden nutzen will. Dabei entspricht das Verfahren dem beim gdm vorgestellten, so daß man selbst die erforderlichen Dateien erzeugen muß.

Ähnlich wie in den alten Versionen des gdm-Pakets benötigt auch der kdm eine PID-Datei mit dem Namen »xdm.pid«, so daß man in der Konfigurationsdatei eine entsprechende Anpassung vornehmen muß. Also sieht die Datei »kdm« im Verzeichnis »/etc/rc.d« folgendermaßen aus:

```
# PROVIDE: kdm
# REQUIRE: DAEMON LOGIN wscons
# KEYWORD: shutdown

. /etc/rc.subr

name="kdm"
rcvar=$name
command="/usr/X11R6/bin/${name}"
pidfile="/var/run/xdm.pid"
required_files="/usr/X11R6/share/kde/config/kdm/kdmrc"
extra_commands="reload"

load_rc_config $name
run_rc_command "$1"
```

Damit der kdm beim Hochfahren automatisch gestartet werden kann, muß die Zeile

```
kdm=YES
```

in die Datei »/etc/rc.conf« eingetragen werden. Zu beachten ist dabei, daß andere Display-Manager auskommentiert werden müssen, da es sonst zu Problemen kommen kann.

Die Konfiguration des kdm kann man mit einem Editor durchführen, da die zugehörige Konfigurationsdatei »/usr/X11R6/share/kde/config/kdm/kdmrc« eine reine Textdatei ist. Viel bequemer ist es aber, das Kontrollzentrum zu diesem Zweck einzusetzen. Die erforderlichen Optionen befinden sich unter dem Punkt *System/Login Manager*.

Soll der kdm für das Einloggen über das Netzwerk (XDMCP) zum Einsatz kommen, muß man wie beim xdm die Dateien »Xaccess« und »xdm-config« im Verzeichnis »/usr/X11R6/lib/X11/xdm« prüfen und gegebenenfalls entsprechend ändern. Doch leider reicht das nicht aus, denn auch in der Konfigurationsdatei des kdm mit dem Namen »/usr/X11R6/share/kde/config/kdm/kdmrc« muß der XDMCP-Dienst explizit freigeschaltet werden. Die entsprechende Option *Enable*, deren Wert man auf *True* setzen muß, befindet sich in der Sektion *{Xdmcp}*. Damit die Änderung wirksam wird, muß der kdm neu gestartet werden, was am einfachsten durch einen Neustart des Rechners oder einen kill-Befehl erfolgt:

```
kill `cat /var/run/xdm.pid`
```

Nach dem *kill* wird – je nach Konfiguration – der kdm entweder automatisch wieder neu gestartet oder durch den Aufruf von *kdm* neu geladen. Es kann allerdings vorkommen, daß man sich dazu erst einloggen muß.

12.15 X im Netz

von Martin Schulze

X11 besticht auch durch seine Netzwerkunterstützung. Im folgenden wird gezeigt, wie man die Ausgabe eines Programms auf einen anderen Bildschirm umleitet.

Die grafischen Ausgaben eines Programms lassen sich mit Hilfe von X11 komplett auf einen anderen Rechner übertragen. Von Textanwendungen ist das vielleicht schon bekannt: Über *telnet*, *rlogin* oder *ssh* logt man sich auf einem anderen Rechner ein, startet dort *vi* oder *mutt* und sieht die Ausgaben auf der lokalen Workstation. Dieses Verhalten läßt sich auch für grafisch orientierte Programme wie beispielsweise GIMP oder *gv* übernehmen. Um die Ausgabe von grafischen Anwendungen umzuleiten, die gegen die normale Xlib gelinkt sind, stehen drei Möglichkeiten zur Verfügung mit folgenden Merkmalen:

- xhost
- xauth
- ssh und xauth

Generell gilt, daß von den Applikationen die Umgebungsvariable *DISPLAY* abgefragt wird. In ihr müssen die Informationen enthalten sein, wo die grafische Ausgabe erfolgen soll. Sie wird in allen drei Fällen mit folgendem Inhalt gesetzt:

```
Rechnername:Display-Nummer.Screen-Nummer
```

Rechnername

Mit *Rechnername* wird der Name des Rechners im Netzwerk bezeichnet, an dem der Monitor physikalisch angeschlossen ist, auf dem die Ausgabe daher erfolgen soll. Hier darf sowohl ein vollständiger Name samt Domain (auch FQDN genannt), ein einfacher Rechnername, wenn die Domain klar ist beziehungsweise in der »/etc/resolv.conf« angegeben wurde, sowie eine IP-Nummer angegeben werden.

Display-Nummer

Als *Display* wird normalerweise eine Menge von Monitoren bezeichnet, die über eine gemeinsame Tastatur und Maus angesprochen werden. Die meisten Workstations haben jedoch nur einen einzigen Monitor angeschlossen, so daß hier meistens ausschließlich der Wert 0 verwendet wird. Es wäre jedoch von der Architektur von X11 her problemlos möglich, an einem Rechner mehrere unterschiedliche Monitore, Tastaturen und Mäuse anzusteuern, auch wenn auf einem herkömmlichen PC die Hardware dies in der Regel nicht hergibt.

Screen-Nummer

Einige Displays teilen sich eine einzige Tastatur und Maus bei mehreren physikalischen Monitoren. Jeder dieser Screens erhält eine eindeutige Nummer, auch hier wird bei 0 zu zählen angefangen. Normalerweise wird hier also auch der Wert 0 verwendet. Bei XFree86 4.x mit Xinerama, mehreren Grafikkarten und Monitoren, könnte auf diese Weise mit 1 der zweite, mit 2 der dritte physikalische Monitor adressiert werden.

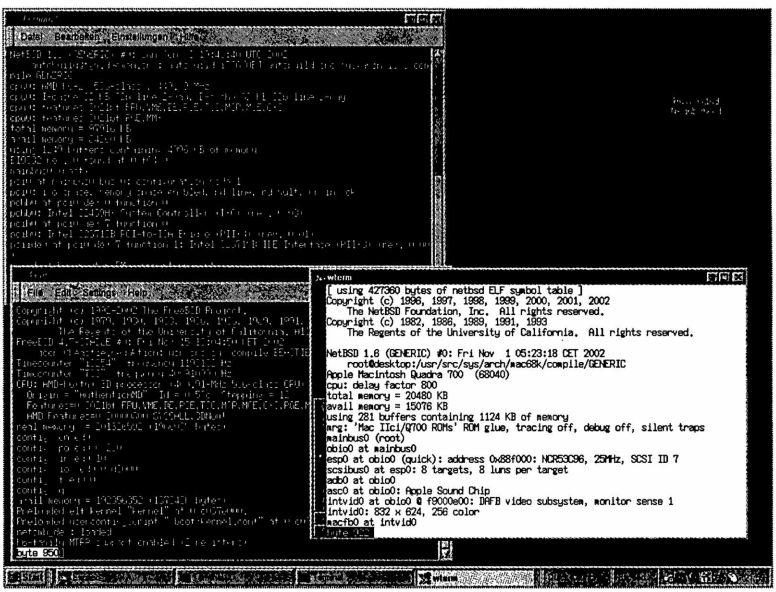


Abb. 12.35: Das Exportieren des Displays ist nicht auf Unix oder gar NetBSD beschränkt. Jeder kann von der Netzwerkfähigkeit von X profitieren, der einen X-Server laufen hat. Hier ist es ein Windows-Rechner auf dem der kommerzielle X-Server Exceed von Hummingbird läuft und auf dem drei X-Fenster auf verschiedenen BSD-Maschinen zu sehen sind.

12.15.1 Ausgabe umlenken

Um die Ausgabe auf ein normales BSD-System umzuleiten, an dem nur eine Tastatur, nur eine Grafikkarte und nur ein Monitor angeschlossen ist, wird die Variable nach der obigen Vorschrift wie folgt gesetzt:

```
DISPLAY=rechnername:0.0
export DISPLAY
```

Alternativ zu dieser Umgebungsvariablen wird von vielen X-Programmen der Parameter *--display dpy* unterstützt, der als Argument die den gleichen Wert erhält, hier also *--display rechnername:0.0*. Ob ein oder zwei Bindestriche beim Parameter anzugeben sind ist übrigens vom Programm abhängig: xterm benötigt nur einen, alle Gnome-Programme zwei. Das muß man einfach ausprobieren.

Die Eingabe muß auf dem Rechner durchgeführt werden, auf dem das Programm gestartet wird, nicht auf dem es ausgegeben werden soll. Zusätzlich dazu muß das System, auf dem die Ausgabe erscheinen soll, so eingestellt werden, daß einem externes Programm seine Ausgaben dorthin schicken darf.

Diese Möglichkeit, die grafische Ausgabe von Programmen umzuleiten, ist recht alt. Früher wurde sie vor allem dafür benutzt, rechenintensive Programme auf dem leistungsfähigsten Server zu starten und die Ausgabe auf die eigene Workstation umzuleiten. Eine Sun-IPX ist nicht unbedingt für die Lösung umfangreicher Differentialgleichungen geeignet. Stehen jedoch Maple oder Mathematica auf einem schnellen Parallelrechner zur Verfügung, wird das Programm dort gestartet, die Ausgabe jedoch auf den eigenen Arbeitsplatz umgeleitet.

12.15.2 Umleitung mit »xhost«

Die Umleiten der Displays mit *xhost* ist die einfachste und älteste Methode. Sie bringt allerdings den Nachteil mit sich, daß bei unvorsichtiger Verwendung jeder Benutzer auf das laufende X11 zugreifen kann, wenn er auf dem entfernten Rechner eingeloggt ist. Damit kann er nicht nur selbst Programme starten, die auf dem X11-Bildschirm erscheinen, sondern zum Beispiel auch die Tastatur verstellen. Daher sollte mit dieser Methode extrem vorsichtig umgegangen werden. Sie wird nur deshalb angegeben, weil dies auf alten Unix-Rechnern die einzige Möglichkeit ist, die Ausgabe umzuleiten.

Mit Hilfe von *xhost* wird auf dem Rechner, auf dem die grafische Oberfläche läuft, das Display für andere Rechner freigegeben. Die Syntax ist recht einfach:

xhost +host	Öffnet das Display für Anwendungen vom angegebenen Host.
xhost -host	Schließt das Display für Anwendungen vom angegebenen Host.

Mit *host* wird in diesem Fall der Rechner (oder dessen IP-Nummer) bezeichnet, auf dem das Programm gestartet wird, dessen Ausgabe umgeleitet werden soll. Als besondere Form darf der Name des Rechners entfallen. In dem Fall darf jeder andere Rechner im Netzwerk auf das Display zugreifen, dort Fenster öffnen, die Konfiguration abfragen oder ändern.

Wenn die Erlaubnis, auf das Display zuzugreifen, mit *xhost* erteilt wird, sollte sie direkt nach dem Start der Anwendung wieder entzogen werden, damit kein anderer Benutzer des entfernten Rechners auf das Display zugreifen kann. Die bereits gestartete Anwendung wird dadurch nicht beeinflußt, da ihr Verbindungskanal bereits geöffnet ist und der Zugang bereits gewährt wurde.

12.15.3 Umleitung mit »auth«

Beim Umleiten der Ausgabe mit *auth* wird mit mit Magic Cookies gearbeitet. Sie werden zwischen dem Server und dem Client ausgetauscht und vom X-Server wie eine Eintrittskarte in die wunderbare Welt der Workstation behandelt. Auf den X-Server dürfen dann nur die Applikationen zugreifen, die das passende Magic Cookie verwenden.

Um dies zu erreichen, wird eine Datei mit Zugriffsrechten und Cookies verwaltet, die sowohl auf dem Client (dem entfernten Rechner) als auch dem Server gepflegt werden muß. Dieses geschieht durch das Programm *xauth*, dessen Bedienung im folgenden kurz erläutert wird. Das Programm extrahiert Informationen aus der Autorisierungsdatei und fügt welche hinzu.

Normalerweise wird die Datei ».Xauthority« verwendet. In einer Umgebung mit einem gemeinsam genutzten Home-Verzeichnis und mehreren X-Servern kann dies zu Problemen führen, da die Cookies für jeden lokalen X-Server neu erzeugt werden. Das würde bedeuten, daß mit dem Start eines neuen X-Servers die Authentifizierung für den ersten gelöscht würde. Damit könnten auf den ersten X-Server keine Ausgaben mehr erfolgen. Dieses Problem wird durch die Umgebungsvariable *XAUTHORITY* umgangen, die den zu verwendenden Dateinamen enthält. So könnte man zum Beispiel immer die Datei ».Xauthority.rechnername« verwenden, indem an entsprechender Stelle diese Variable gesetzt wird. Den Rechnernamen erhält man durch Verwendung von `` `hostname` ``, also zum Beispiel durch das folgendes Fragment:

```
XAUTHORITY=~/.Xauthority.`hostname`
export XAUTHORITY
```

Um mit diesem Mechanismus zu arbeiten, muß beim Start von X11 dem X-Server mitgeteilt werden, daß die Authentifizierung über *xauth* geschehen soll. Normalerweise geschieht das bereits automatisch beim Aufruf von *startx* und beim grafischen Login über *xdm*, *gdm* oder *kdm*.

Wenn die Datei »~/.Xauthority« vorhanden ist und einen aktuellen Zeitstempel trägt, ist das System entsprechend vorbereitet und verwendet bereits diese Art der Authentifizierung.

Ist das jedoch nicht der Fall, wird wie folgt vorgegangen: Der X-Server wird nicht mit *startx* gestartet, sondern direkt mit dem Programm *xinit*. Allerdings wird vorher die Authentifizierungsdatei angepaßt beziehungsweise angelegt und als Parameter übergeben. Die folgenden Befehle sind dazu nötig:

```
[1] XAUTHORITY=~/.Xauthority
[2] export XAUTHORITY
[3] rm -f $XAUTHORITY
[4] xauth add :0 . `mcookie`
[5] xauth add `hostname`:0 . `mcookie`
[6] xinit -- -auth $XAUTHORITY
```

Zuerst wird in der ersten Zeile die Variable *XAUTHORITY* sinnvoll gesetzt, damit später auf dessen Inhalt zugegriffen werden kann. In der zweiten Zeile wird die Variable exportiert. In der dritten Zeile wird eine bereits vorhandene Autorisierungsdatei gelöscht.

In Zeile 4 wird die Datei erstellt und ein Cookie für den Unix-Domain-Socket hinzugefügt. Mit *:0* wird die normale Verbindung für grafische Applikationen beschrieben. Das geht nicht über eine normale Netzwerkverbindung, sondern über eine rechner-interne Verbindung.

In Zeile 5 wird ein weiterer Cookie hinzugefügt und zwar für *rechnername:0*.

In der sechsten und letzten Zeile wird der X-Server gestartet. Die auf »--« folgenden Argumente werden direkt an X übergeben. Damit wird dem X-Server mitgeteilt, wie die Authentifizierung zu erfolgen hat.

Damit sind die Grundlagen gelegt, um auch von einem entfernten Rechner auf den X-Server zuzugreifen. Gestattet ist das jedoch noch längst nicht. Um das zu ermöglichen, muß das passende Cookie auf den zweiten Rechner übertragen werden. Dazu wird zuerst das entsprechende Cookie aus der Datei extrahiert, auf den entfernten Rechner übertragen, dort eingebaut, das Display gesetzt und das Programm gestartet. Zusammen ergibt das folgende Schritte:

```
[1] xauth extract - `hostname`:0 > cookie
[2] rcp cookie remote:
[3] rlogin remote
[4] xauth merge - <\<> cookie
[5] DISPLAY=rechner:0.0; export DISPLAY
[6] xterm
```

In der ersten Zeile wird der Cookie extrahiert und in der Datei »cookie« gespeichert.

In der zweiten Zeile wird die Datei mit dem Cookie auf den entfernten Rechner übertragen, hier könnten genauso auch *ftp* oder andere Übertragungsdienste verwendet werden.

In Zeile 3 erfolgt das Einloggen auf dem entfernten Rechner.

In Zeile 4 wird nun der Cookie in die lokale Authentifizierungsdatei übernommen und in Zeile 5 das Display passend gesetzt.

Zum Schluß wird in der sechsten Zeile die Anwendung gestartet. Wenn alles geklappt hat, erscheint die Ausgabe jetzt auf der lokalen Workstation.

Als Kurzform der Zeilen 1 bis 5 wird der folgende Einzeiler (hier aus Platzgründen aufgeteilt) verwendet:

```
xauth extract - `hostname`:0 | \
  rsh remote xauth merge -
```

12.15.4 Umleitung mit »ssh«

Viel einfacher gestaltet sich die Umleitung von grafischen Programmen, wenn *ssh* und *xauth* installiert sind. In diesem Fall kann nämlich das gesamte Vorgehen von der Secure Shell übernommen werden. Dazu wird die Option *-X* angegeben. Die Secure Shell tunnelt dann eine X11-Verbindung über den verschlüsselten Kanal, der zum entfernten Rechner aufgebaut wurde. Auf der anderen Seite, auf dem entfernten Rechner, wird das Display entsprechend gesetzt. Die *ssh* fängt dort die Ausgaben ab, tunnelt sie durch die SSH-Verbindung zu dem Rechner, auf dem die grafische Oberfläche läuft, und der lokale X-Server stellt die Fenster dar.

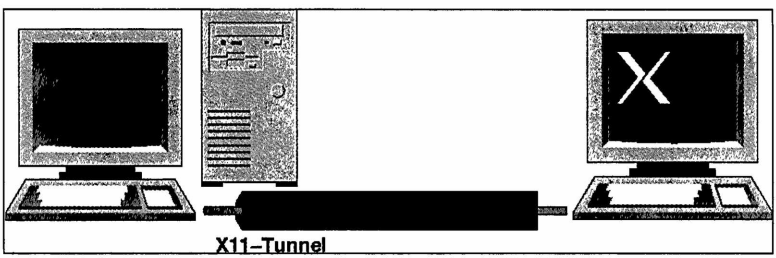


Abb. 12.36: Die X11-Verbindung wird in der SSH-Verbindung getunnelt

Zusätzlich muß eine Weiterleitung von X-Anwendungen erlaubt sein. In der »/etc/ssh/sshd_config« muß folgendes ebenfalls eingetragen sein:

```
X11Forwarding yes
```

Dieses Vorgehen bietet mehrere Vorteile. Die Verbindung und gesamte Übertragung erfolgt im Normalfall verschlüsselt. Ein Mithören der Verbindung ist dadurch äußerst schwierig. Was bei einer herkömmlichen X11-Übertragung problemlos möglich ist – die IP-Pakete abzuhören und die Ausgabe auf einem zweiten Monitor parallel darzustellen – wird dadurch erheblich erschwert.

Ein weiterer Vorteil liegt darin, daß keine zusätzlichen Ports auf dem Server erforderlich sind. Somit kann man das X11-Forward auch durch eine Firewall betreiben, die restriktiv nur wenige Ports eingehend durchläßt. Wenn eine ausgehende SSH-Verbindung gestattet ist, ist das ausreichend, da die X11-Verbindung dadurch getunnelt wird.

12.16 Alternative automatische X-Konfiguration

von Jörg Braun

Die Konfiguration von XFree86 4 mit den Werkzeugen die die grafische Oberfläche mitliefert, egal ob das *xf86config*, *xf86cfg* oder auch *XFree86 -configure* ist, ist mühsam. Sie können es auch anders probieren. Dazu kopieren Sie sich (als User *root*) von der dem Buch mitgelieferten CD-ROM die Dateien »mkxf86config« und »XF86Config-4.in« aus

dem Verzeichnis »/etc/X11« in das gleichnamige Verzeichnis auf der Festplatte. Beide Dateien müssen leicht modifiziert werden. In der »XF86Config-4.in« ist der Bereich der Tastatur auskommentiert. Ändern Sie die folgenden Zeilen von

```
# Section "InputDevice"
#       Identifier  "Keyboard0"
#       Driver      "keyboard"
#        Option      "CoreKeyboard"
#       Option "XkbRules" "xfree86"
#       Option "XkbModel" "pc104"
#       Option "XkbLayout" "us"
#       Option "XkbVariant" "nodeadkeys"
# EndSection
```

in

```
Section "InputDevice"
      Identifier          "Keyboard0"
      Driver              "keyboard"
      Option              "CoreKeyboard"
      Option "XkbRules"   "xfree86"
      Option "XkbModel"   "pc105"
      Option "XkbLayout"  "de"
      Option "XkbVariant" "nodeadkeys"
EndSection
```

und speichern Sie die Datei.

Nun müssen Sie in der Skriptdatei ebenfalls einige Zeilen hinzufügen. Zwischen:

```
### Utility Function(s)
# Reread boot command line; echo last parameter's argument ...
getbootparam(){
read CMDLINE </proc/cmdline 2>/dev/null
case "$CMDLINE" in *$1=*) ;; *) return 1; ;; esac
result="${CMDLINE##*$1=}"
result="${result%%[     ]*}"
echo "$result"
return 0
}
```

und den Zeilen

```
# Read in what hwsetup has found for X
[ -f /etc/X11/xserver ] && . /etc/X11/xserver
```

fügen Sie die folgenden Zeilen ein:

```
# New for NetBSD configuration
/usr/X11R6/bin/XFree86 -configure   # 2> /dev/null
echo -n "XMODULE=" > /etc/X11/xserver
grep "BoardName" /root/XF86Config.new | cut -d '"' -f 2 | \
     cut -d ' ' -f 1 >> /etc/X11/xserver
echo -n "XDESC=" >> /etc/X11/xserver
grep "BoardName" /root/XF86Config.new | cut -d '"' -f 2 | \
     cut -d ' ' -f 1 >> /etc/X11/xserver
rm -f /root/XF86Config.new
```

Außerdem müssen Sie alle auf »/etc/knoppix« lautenden Einträge in der Datei auf »/etc/X11« ändern.

```
        i128
        ati
        i810
        ark
        cyrix
        siliconmotion
        vesa
        vga
(++) Using config file: "/root/XF86Config.nee"

XFree86 detected your mouse at device /dev/wsmouse0.
Please check you config file if the mouse is still not
operational, as by default XFree86 tries to autodetect
the protocol.

Your XF86Config file ist /root/XF86Config.new

To test the server, run 'XFree86 -xf86config /root/XF86Config.new'

 Video is Mach64, using XFree86(Mach64) Server
 Monitor is Generic Monitor
 Using Modes "1024x768" "800x600" "640x480"
```

Abb. 12.37: Die Erkennung der Grafikkarte hat geklappt

Falls sich im Verzeichnis bereits eine Datei mit dem Namen »XF86Config« oder »XF86Config-4« befindet, löschen Sie sie und rufen Sie in »/etc/X11« das geänderte Skript mit

```
./mkxf86config
```

auf. Nun wird eine brauchbare Konfigurationsdatei für X angelegt. Prüfen Sie dabei, ob die Grafikkarte gefunden wurde und nicht nur ein VESA-Modus verwendet wird, der zwar auch einsetzbar, aber doch langsamer ist als ein an die Karte anpaßter Server. Sie sehen die verwendeten Parameter an der Ausschrift des Skripts in Abb. 12.37.

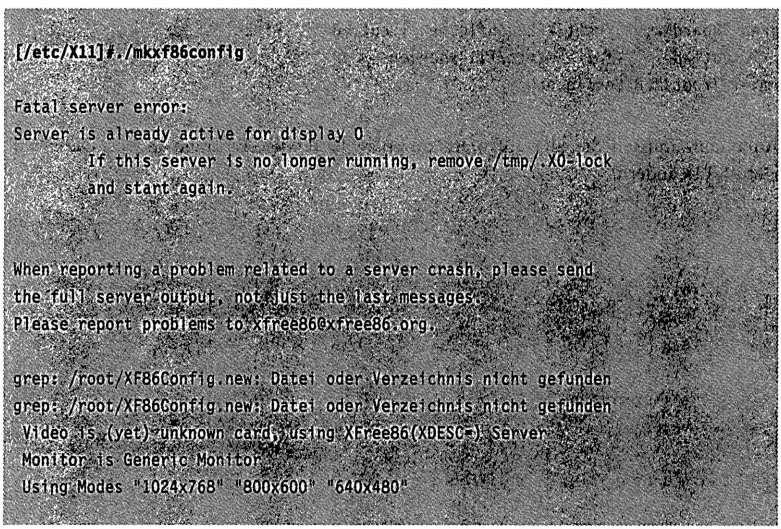


Abb. 12.38: Eine X-Erkennung unter X muß fehlschlagen!

Hierbei muß natürlich die erkannte Grafikkarte die Ihres System sein. Wenn Sie die Anzeige in Abbildung 12.38 erhalten, haben Sie das Skript aus einem X-Terminal gestartet. Das geht natürlich nicht! Das Skript muß auf einer echten Textkonsole ausgeführt werden.

Sie sollten nun die soeben erzeugte X-Konfigurationsdatei von »XF86Config-4« in »XF86Config« umbenennen und versuchsweise *startx* aufrufen.

13 Drucken mit NetBSD

13.1 Drucken mit dem lpd

von Thorsten Lindloff

Wie bei allen Unix-Betriebssystemen gibt es auch für NetBSD ein netzwerkfähiges Drucksystem. Der Name *lpd* steht kurz und knapp als Abkürzung für »Line Printer Daemon«.

Gestartet wird der *lpd* im Regelfall beim Booten, außer Sie haben etwas an seiner Konfiguration geändert oder setzen ihn zum ersten Mal auf. Dann starten Sie *lpd* mit einem

```
# /etc/rc.d/lpd start
```

oder

```
# /etc/rc.d/lpd stop
# /etc/rc.d/lpd start
```

Bevor Sie den Druckerdaemon verwenden können, müssen Sie zuerst einmal mit einem Editor eine Änderung in der »/etc/rc.conf« vornehmen. Suchen Sie die Zeile *start_lpd* und setzen Sie die Variable von *no* auf *yes*. Dann wird der *lpd* wie oben beschrieben gestartet. Fehlt die Zeile, müssen Sie sie eintragen.

Wenn der *lpd* gestartet wird, durchsucht er die Datei »/etc/printcap« nach vorhandenen Druckern. Anschließend werden nach einem eventuellem Systemcrash übriggebliebene Druckaufträge abgearbeitet. Danach wartet er auf neue Aufträge. Das geschieht über die sogenannten Systemaufrufe *listen* und *accept* (zu Deutsch: horchen und akzeptieren).

Wenn ein Auftrag eingeht, werden die Aufträge in die Warteschlange eingereiht, in den Spoolbereich zwischengelagert und anschließend weiterverarbeitet. Für jeden dieser Aufträge wird ein Kindprozeß gestartet, damit der »elterliche« lpd-Prozeß weiter nach neuen Aufträgen Ausschau halten kann.

Der Zugriff auf dieses System muß geregelt sein. Wenn Sie den Drucker mehreren Maschinen im Netz zur Verfügung stellen wollen, muß das zuerst einmal erlaubt sein. Das geschieht entweder in der »hosts.equiv« oder in der »hosts.lpd«. Beide Verfahren haben ihre Vor- und Nachteile. In »hosts.lpd« steht nur, welcher Host auf dieser Maschine drucken darf. »hosts.equiv« entspricht funktionell in etwa der Datei ».rhosts«, die in jedem Benutzerverzeichnis angelegt wird, wenn ein neues Benutzerkonto eingerichtet wird. Der Unterschied ist, daß diese Datei eben nicht nur für einen Benutzer gültig ist, sondern für alle Benutzer auf den Maschinen, die hier eingetragen sind. Diese Datei verlangt allerdings danach, daß Sie den Benutzern auf diesen Hosts auch vertrauen können. Der Vorteil ist, daß es sehr komfortabel ist, mit dieser Datei zu arbeiten. Sie sollten dabei aber auch im Auge behalten, daß Ihre User auch nur Menschen sind und daß gerne mal ein paar Fehler einschleichen, die sich schnell fatal auswirken können.

Deshalb: Wenn Sie mit dem Rechner ausschließlich drucken wollen oder eine sichere Umgebung brauchen, sollten Sie nur »hosts.lpd« einsetzen.

Das Berkeley Line Printer Spooling System, kurz *lpd* genannt, übernimmt in Ihrem System diese Aufgaben:

- Der Druckdaemon regelt den Zugriff auf den lokalen Drucker und leitet Daten an einen anderen Drucker im Netzwerk weiter, wenn kein eigener Drucker vorhanden ist.
- Er verhindert, daß gleichzeitig mehrere Benutzer auf den Drucker zugreifen, indem er eine Queue (Warteschlange) für jeden Drucker einrichtet, für den er konfiguriert ist.
- Er fügt Leer- oder Bannerseiten ein, damit der einzelne Benutzer seine Seiten in einem großen Stapel Papier wiederfindet.
- Er regelt die Kommunikationsparameter für Drucker, die an die seriellen Schnittstellen angeschlossen sind.
- *lpd* arbeitet mit druckerspezifischen Kommandos, um einen Drucker für verschiedene Druckersprachen einzurichten oder um überhaupt einen lesbaren Ausdruck zu bekommen.

Wenn Sie einen Drucker besitzen, der fähig ist, entweder PCL5 und höher zu verarbeiten oder wenn es sich um ein Tintenstrahl-Modell von Canon, Epson oder Hewlett Packard handelt, sind Sie meistens schon recht gut ausgerüstet. Allerdings gibt es auch unter diesen Modellen ein paar Geräte, die mit dem lpd nicht funktionieren. Bei HP sind das die Modelle 710, 720 und 820; bei Epson sind die Modelle der »Stylus Color Photo«-Baureihe dafür bekannt; bei den Lexmark-Modellen handelt es sich um die Tintenstrahldrucker der Z-Serie. Grundsätzlich können Sie davon ausgehen, daß wenn auf einem Drucker der Aufkleber »for Windows« prangt, das bedeutet: »For Windows only«. Wenn Sie einen neuen Drucker kaufen wollen, lassen Sie diese Geräte bitte im Laden. Diese Komponenten drucken über eine Software-Schnittstelle, die Graphics Device Interface genannt wird. Sie können sie zumindest nicht im Zusammenhang mit dem lpd verwenden, weil sie »dumm« sind und die Rechenarbeit einer Software und damit dem Prozessor des Computers überlassen. Dazu kommt noch der Umstand, daß die Hersteller von Hardware die Entwickler von Unix, die sich um solche Treiber kümmern, nicht immer unterstützen. Meistens geschieht das unter dem Deckmantel des Verbraucherschutzes. Ich hatte vor meinen Linux- und NetBSD-Zeiten ein paar Jahre lang eine solche Zitrone, die nicht gerade zuverlässig war und sich nicht für Unix-Betriebssysteme eignete. Dazu muß man auch hier sagen, daß das neuere CUPS diese Drucker besser unterstützt und daß es bereits einen Ghostscript-Treiber für solche Drucker gibt.

13.1.1 Ghostscript

Sie brauchen im Gegensatz zu frühen Unix-Versionen heute keinen teueren Postscript-Drucker mehr. Falls der Drucker postscriptfähig ist, ist das aber auch kein Schaden. Das Übersetzen in die »Sprache« des jeweiligen Druckers übernimmt in NetBSD wie bei allen freien Unix-Systemen das Programm Ghostscript.

Die Herausforderung besteht darin, dieses doch schon recht betagte Programm, das es immerhin auch schon seit Anfang der neunziger Jahre gibt und das ursprünglich für die alten Zeilendrucker konzipiert wurde, zu einer zufriedenstellenden Arbeit zu bewegen. Wir werden uns hier auf zwei Drucker-Standardexemplare konzentrieren: Da gibt es einmal den besagten PCL5-Drucker, der in den meisten Fällen mit dem HP Laserjet 4 kompatibel ist oder bei dem es sich sogar um einen Original HP-Laserjet handelt, und eines der weit verbreiteten Tintenstrahl-Modelle. Getestet wurde mit einem Samsung ML5000A-Laserdrucker und einem HP Deskjet 610c. Beide sind Drucker, die problemlos mit dem lpd laufen.

Wenn Sie einen neuen Laserdrucker kaufen wollen, bitten Sie den Verkäufer darum, einen Blick in die Gebrauchsanleitung und die technischen Daten werfen zu dürfen. Es sollte in den technischen Daten ein Hinweis auftauchen, welcher Prozessor das Gerät steuert und welche Sprache der Drucker versteht. Alles ab PCL5 sollte funktionieren.

13.1.2 Der Druckvorgang

Wenn der Computer bootet, wird der lpd gestartet. Dieser Dienst untersucht die Datei »/etc/printcap« nach Druckern, für die besagte Warteschlangen verwaltet werden sollen. Wenn jemand *lpr* aufruft, verbindet sich dieses Programm über den Socket »/dev/printer« mit dem lpd und teilt ihm mit, was wie auszudrucken ist.

In diesem Kapitel geht es um den Regelfall, also um Drucker, die am PC-Parallelport angeschlossen werden. Tabelle 13.1 zeigt die grundsätzlichen Druckbefehle. Man sollte sie auch dann kennen, wenn man später mit einem Office-Paket wie beispielsweise OpenOffice arbeiten will.

Befehl	Funktion
lpd	Das ist der eigentliche Daemon. Er wird beim Systemstart hochgefahren und steuert die Druckprozesse.
lpr	Damit werden die Druckjobs an die Warteschlange des lpd weitergereicht.
lpq	Zeigt an, welche Druckaufträge in der Warteschlange stehen.
lpc	Mit diesem Befehl werden die Warteschlangen gestoppt, gestartet, sortiert und so weiter .
lprm	Damit entfernt man einen Auftrag aus der Warteschlange.

Tabelle 13.1: Die wichtigsten Druckbefehle

Zuerst muß man sicherstellen, daß der Druckeranschluß und der Drucker selbst vom Kernel erkannt werden. Dazu prüft man den Output von *dmesg*. Dort sollte eine Zeile auftauchen, die der folgenden Ausschrift ähnlich ist:

```
lpt0 at isa0 port 0x378-0x37h irq 7
```

Wenn das der Fall ist, können Sie drucken, falls nicht, wird ein anderer Kernel benötigt. Wird der GENERIC-Kernel eingesetzt und die Controller-Hardware verwendet diese Adresse, ist alles klar.

Jetzt muß der automatische Start des Druckdaemons in der Datei »/etc/rc.conf« mit der folgenden Zeile aktiviert werden:

```
lpd=YES
```

Anschließend läßt sich der Daemon mit einem Aufruf von

```
# lpd -s
```

starten. Die Prüfung, ob der Daemon tatsächlich gestartet wurde, erfolgt über

```
# ps ax | grep lpd
```

Dann sollte man einen Test fahren:

```
# lptest 10 20 /dev/lpt0
```

Wenn Sie wissen wollen, wie das Ergebnis dieses Testlaufs aussehen soll, können Sie das Ergebnis auch ohne Drucker begutachten:

```
# lptest 10 20
```

Wenn Sie einen Testlauf gefahren haben, werden Sie wahrscheinlich feststellen, daß der ausgedruckte Text folgendermaßen aussieht:

```
Das ist
      ein
         Treppeneffekt.
```

Dieses kleine Problem hat seine Ursache darin, daß ein Drucker für PC-Systeme ein LF/CR-Signal (Line Feed/Cariage Return) braucht, um eine Zeile weiterzugehen und einem Unix-Drucker ein karges LF-Signal ausreicht, um eine neue Zeile anzufangen.

Man nennt das auch einen Staircase-Effekt. Dagegen kann man aber etwas tun. Man muß die beiden folgenden Dateien entweder schon auf dem Rechner haben oder neu anlegen:

```
/etc/printcap
/etc/hosts.lpd
```

In der »printcap« steht die eigentliche Konfiguration des Druckers.

Beginnen wir mit der einfachsten Variante, einem Drucker, der sich als HP Deskjet 500 ansprechen läßt. Der billige HP paßt in diese Kategorie.

Sie müssen mehrere Dinge tun:

- Der Drucker braucht einen Namen. Nennen Sie ihn *lp*, wie es beim lpd ohnehin Standard ist.
- Sie brauchen ein Spool-Verzeichnis unter »/var/spool/lpd«. Legen Sie es an:

  ```
  mkdir /var/spool/lpd/lp
  chmod 755 /var/spool/lpd/lp
  ```

  Vergessen Sie auch nicht, die Zugriffsrechte für die Dateien zu überprüfen. Im Normalfall werden sie richtig gesetzt, wenn man einen fertigen Daemon verwendet. Eine Prüfung schadet dennoch nicht.

  ```
  -r-sr-sr-x  1  root  lp    /usr/bin/lpr
  -t-sr-sr-x  1  root  lp    /usr/bin/lprm
  -rwxr--r--  1  root  root  /usr/sbin/lpd
  -r-xr-sr-x  1  root  lp    /usr/sbin/lpc
  drwxrwxr-x  1  root  lp    /var/spool/lpd/
  drwxr-xr-x  1  root  lp    /var/spool/lpd/lp
  ```

- Der Drucker muß in die »/etc/printcap eintragen werden:

  ```
  # Deskjet 500 lokal
  lp|dj|deskjet
       :sd=/var/spool/lpd/lp:\
       :mx#0:\
       :lp=/dev/lpt0:\
       :sh:
  ```

Das, was oben zu sehen ist, wird »Strophe« genannt. Eigentlich können Sie ein solche Strophe auch in einer einzigen Zeile unterbringen. Das dient aber nicht der Lesbarkeit einer solchen Datei. Deshalb muß ein Zeichen als Zeilenverbinder angegeben werden, das dem *lpd* sagt, daß hier noch nicht das Ende ist: das ist die Aufgabe des Backslashes am Ende der Zeilen. Um die Befehle voneinander zu trennen, kommt der Doppelpunkt zum Einsatz.

Eintrag	Beschreibung
lp|dj|deskjet	Der Druckername und seine Aliasnamen.
:sd=/var/spool/lpd/lp:	Das Spool-Verzeichnis.
:lp=/dev/lpt0:	Die Anschluß-Schnittstelle (erster Parallelport).
:mx#0:	Maximale Auftragsgröße, hier 0 für »keine Begrenzung«.
:sh:	Bitte kein Deckblatt mit ausgeben.

Tabelle 13.2: Eine einfache printcap

Wenn jetzt ein Probedruck gemacht wird, stellt man fest, daß der Ausdruck immer noch den Treppeneffekt zeigt. Wenn Sie ein einfache Unix-Textdatei an den Drucker schicken, bekommen Sie eine Ausgabe wie oben bereits beschrieben. Würden Sie jetzt versuchen, eine Postscript-Datei auszudrucken, würde der Drucker das sogar tun, aber auf die Ausgabe der Postscript-Befehle nicht verzichten. Sie erhalten dann noch mehr als den oben gezeigten Kauderwelsch. Um dieses Problem zu umgehen, ist eine weitere Maßnahme nötig: ein Filter. Dazu kann man in der »printcap« zwei Attribute setzen: *if* (Inputfilter) und *of* (Outputfilter). In unserem Fall ist ein *if* richtig.

Die neue »printcap« sieht dann folgendermaßen aus:

```
# Deskjet 500 lokal
lp|dj|deskjet
        :sd=/var/spool/lpd/dj:\
        :mx#0:\
        :lp=/dev/lpt0:\
        :if=/var/spool/lpd/deskjet/filter:\
        :sh:
```

Diese Datei löst sich nach dem in Tabelle 13.3 gezeigten Schema auf. Das ist eine sehr gute Warteschlange, die sich zum Ausdrucken von Textdateien hervorragend eignet. Jetzt bleibt aber immer noch das Problem, daß niemand nur Texte drucken möchte. Meistens sollen Postscript- oder Grafikdaten, manchmal auch Manpages ausgegeben werden. Das wird mit einem Inputfilter erledigt. Man muß eigentlich nur den bereits vorhandenen Zeilenumbruchfilter um ein paar Befehle erweitern.

Wird ein Filter um so ein Feature erweitert, hat man das, was sich »Magicfilter« nennt.

Solche Filter gibt es für einige Drucker bereits fertig. Eines dieser Pakete heißt *Magicfilter*, das andere *apsfilter*. Für apsfilter kommt mit der Distribution ein Konfigurationstool. Magicfilter muß standardmäßig manuell konfiguriert werden, außer man besitzt ein Tool, wie das, das der Debian-Linuxdistribution beiliegt und dort standardmäßig installiert wird. Eine NetBSD-Variante kann unter *http://www.lindloff.com/netbsd/magicfilter.html* heruntergeladen werden. Das Tool funktioniert, läuft aber interaktiv auf der Konsole. Was die Druckqualität am Ende angeht, geben sich beide Filtertechniken nichts. Sie arbeiten immer mit denselben Programmen zusammen. Ich werde mich an dieser Stelle allerdings primär auf den Magicfilter konzentrieren. Dieser Konfigurator kann zwar

nicht allzu viel, wird Ihnen aber zumindest zu einer »printcap« verhelfen, aus deren Lektüre Ihnen die Syntax dieser Datei klar werden sollte.

Eintrag	Funktion
lp|dj|deskjet :	Bezeichnet den Namen des Druckers, den Sie, wenn Sie wollen, auch »Karlheinz« nennen können. Sie haben keine Beschränkungen.
sd	Steht für »Spool Directory«. Hier wird das Verzeichnis eingetragen, in dem der lpd die auszudruckenden Daten zwischenlagert, bis sie an der Reihe sind. Genaugenommen ist das eine Warteschlange. In diesem Fall ist das das Verzeichnis »/var/spool/lpd/lp«.
mx#0	Hier wird festgelegt, wie groß die Datei maximal sein darf. Wenn hier eine »0« steht, gibt es keine Beschränkungen.
if	Das bezeichnet den Namen des Inputfilters, den Sie auf einem PC-Drucker meistens benötigen, um dem Staircase-Problem aus dem Weg zu gehen. Hier befindet er sich unter »/var/spool/lpd/lpfilter«.
sh	Besagt in diesem Fall, daß kein Deckblatt mit ausgegeben soll. Sie besitzen beispielsweise ein kleines Netzwerk mit zwei Rechnern, bei dem das nicht notwendig ist.

filter	Ist der Name des Filters. In diesem Fall ist er in Perl geschrieben, er ist sehr einfach gehalten:
	`#!perl`
	`while (<stdin>) { chop $_; print ''$_\r\n''; };`
	`print''\f'';`

Tabelle 13.3: Die Einträge der Datei »printcap«

Magicfilter hat gegenüber dem apsfilter einen kleinen, aber gravierenden Nachteil: Er kann nur mit Postscript-Daten etwas anfangen. Weil die meisten Daten aber nicht in Postscript formatiert werden, muß man die Daten zuerst nach Postscript konvertieren. Für diesen Zweck gibt es in »/usr/pkg/libexec« ein Verzeichnis namens »magicfilter«, in dem es bereits vorgefertigte Filter für verschiedene Drucker gibt.

13.1.3 Magicfilter installieren

Installiert man Magicfilter von der CD, werden noch einige Pakete nachinstalliert:

Ghostscript	Der Postscript-Interpreter.
TeX	Das ist ein Satzsystem, das ursprünglich für den Buchdruck entwickelt wurde, sehr gut und performant ist und aus dessen Output sich ohne allzu großen Aufwand auch PDF-Dateien erzeugen lassen. Mit TeX werden auch einige Formatierer installiert, die verschiedene Bildformate in Postscript konvertieren.

Eine Anmerkung zur Sicherheit

Postscript ist eine voll funktionale Sprache, die unter anderem auch den Aufruf von Programmen erlaubt. Hier drohen Sicherheitsprobleme, weil der lpd mit root-Rechten läuft.

Nun arbeiten wir mit dem Magicfilter. In der »printcap« ändert sich nur eine Zeile, nämlich die, in der steht, welcher Filter verwendet werden soll:

```
# Deskjet 500 lokal
lp|dj|deskjet
        :sd=/var/spool/lpd/dj:\
        :mx#0:pl#66:pw#70:\
        :lp=/dev/lpt0:\
        :if=/usr/pkg/libexec/magicfilter/dj500-filter:\
        :sh:
```

Man sollte des weiteren festlegen, wie groß die ausgedruckte Seite sein soll. Die Angaben können in Zeilen und Zeichen oder auch in Punkten erfolgen. Im obigen Beispiel erfolgte die Definition in Zeilen und Zeichen.

pl	Steht für »page length«, also der Seitenlänge in Zeilen.
pw	Steht für »page width«, der Seitenbreite in Zeichen. Hier kann man gegebenenfalls auch noch herumprobieren.

Wenn jetzt ein Ausdruck angefertigt wird, sieht der Text so aus:

```
Das ist kein Treppeneffekt.
```

13.1.4 Drucken im Unix-Netz

Gesetzt dem Fall, Sie wollen Ihren Drucker mit mehreren Maschinen einsetzen, was ja Kosten spart, kommt Ihnen der *lpd* natürlich auch entgegen. In einem solchen Fall wird auf dem Client für einen HP Laserjet4 eine »printcap« angelegt, die folgendermaßen aussieht:

```
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :rm=192.168.1.7:\
        :rp=hplj41:\
        :lpd_bounce=true:\
        :if=/usr/pkg/libexec/magicfilter/ljet4-filter:
```

Eintrag	Beschreibung
lp:\	Name der Warteschleife auf dem lokalen Rechner.
:sd=/var/spool/lpd/lp:\	Die lokale Warteschlange.
:rm=192.168.1.7:\	Auf dieser Maschine soll gedruckt werden.
:rp=hplj4l:\	Die Warteschlange auf dem entfernten Rechner.
:lpd_bounce=true:\	Wenn etwas nicht ordnungsgemäß funktioniert, geht der Auftrag zurück.
if=/usr/pkg/libexec/\ magicfilter/ljet4-filter:	An dieser Stelle befindet sich der lokale Druckfilter.

Tabelle 13.4: printcap-Einträge bei einem Netzwerkdrucker

Nun ist die »printcap« eigentlich fertig eingerichtet. Das reicht allerdings noch nicht. Dem Server, der ja nur das macht, was man ihm sagt, muß mitgeteilt werden, wer denn nun überhaupt drucken darf.

Das geschieht am besten in der bereits eingangs erwähnten Datei »/etc/hosts.lpd«. Dort werden entweder die Rechnernamen oder die IP-Adressen der Maschinen eintragen, ganz nach persönlichem Geschmack. Die Syntax ist: Eine Zeile = ein Rechner.

Ein Sonderfall liegt vor, wenn man entweder einen GDI-Drucker besitzt oder wenn Windows-Rechner im Netzwerk stehen, die ebenfalls auf der NetBSD-Maschine drukken sollen. Damit das überhaupt geht, müssen vorab mehrere Entscheidungen getroffen werden:

- Soll der NetBSD-Rechner auch als Windows-Fileserver mit Samba dienen?
- Ist der verwendete Drucker ein GDI-Modell?
- Befinden sich unter Umständen mehrere Drucker im Netzwerk?
- Oder will man nur von einem Windows-Rechner aus auf der NetBSD-Maschine drucken?

Fall 1: Ein Windows-Rechner druckt auf einem NetBSD-Rechner

Der letzte Fall ist am einfachsten zu lösen.

Wenn Windows NT, 2000 oder XP installiert ist, ist die Einrichtung trivial: Die Drukkereinstellungen werden nach dem im folgenden beschriebenen Schema geändert (ich gehe bei meinen Ausführungen davon aus, daß der bisherige Drucker an die NetBSD-Maschine angeschlossen werden soll). Auf diesen Betriebssystemen kann ein *lpd* nach Unix-Standard nachinstalliert werden. Dazu ruft man im Startmenü von Windows (hier beschrieben unter NT 4) das Feld *Einstellungen* auf, darin *Drucker*. Klicken Sie das Symbol für Ihren Drucker mit der rechten Taste an und wählen Sie *Eigenschaften*. Im dort erscheinenden Register klicken Sie auf die Karte *Anschlüsse* und dort auf *Hinzufügen*. Eine Auswahlliste erscheint, dort doppelklicken Sie *LPR Drucker*. Dann werden Sie nach der IP-Adresse und dem Namen des Druckers gefragt, Sie geben beides an (der Drucker-

name ist immer noch *lp*). Anschließend legen Sie die Windows-NT-Installations-CD ein. Jetzt wird ein *lpd* nach Unix-Muster auf dem System installiert. Unter Windows 2000 und XP ist die Verfahrensweise ähnlich. Anschließend starten Sie das System neu.

Anders gelagert ist die Windows-9x-Schiene inklusive der Millenium-Version. Diese Varianten sind eher für den Privatanwender gedacht und besitzen keinen Unix-Druckertreiber. Hier hat man die Wahl, sich entweder einen freien Client aus dem Internet zu besorgen oder diesen Job mit Hilfe von Samba zu erledingen. Die bequemere Variante ist nicht etwa der Einsatz von Samba als Druckerserver, sondern die hier vorgestellte, insbesondere, wenn man die Dienste eines Samba-Servers ansonsten nicht braucht. Es gibt für diesen Zweck ein kleines Windows-Programm namens *pmlpr*, das man bei der Uni Kiel herunterladen kann und das hervorragend als Spooler funktioniert. Die Download-Adresse ist *http://www.uni-kiel.de/rz/ausgabe/pmlpr/pmlpr.html*.

Das Programm wird installiert, indem man nach dem Download die Datei einfach anklickt und bei der Installation durchgehend mit »ok« bestätigt. Anschließend wird es nach folgendem Muster eingerichtet:

- Sie werden nach dem Ordner gefragt, in dem das Tool abgelegt werden soll, das ist in der Voreinstellung »c:\pmlpr«. Sollten Sie wider Erwarten bereits eine ältere Version des Programms installiert haben, können Sie die bereits vorhandenen Daten sichern.
- Dann wird ein Fenster mit der Frage nach dem Spool-Verzeichnis angezeigt. Das nennen Sie »c:\pmspool« (Defaultwert). Wenn das getan ist, geben Sie im nächsten Fenster unter *Your's PC name* Ihre IP-Adresse oder den Rechnernamen und in *Your name* den Benutzernamen ein.
- Im nächsten Fenster werden Sie gefragt, in welche Programmgruppe das *pmlpr*-Icon geschrieben werden soll. Soll das Programm automatisch gestartet werden, ist das *Autostart*.
- Das Fenster *Extra Information* weist Sie auf eine Datei »readme.txt« hin, die Sie sich anschauen sollten. Im letzten Fenster (*What to do next*) werden die weiteren Schritte zur Konfiguration erklärt. Sie ist denkbar einfach und bedarf eigentlich keiner weiteren Erklärung.

pmlpr hat allerdings einen Nachteil: Es steht im Ruf, bei Dateien, die größer als 20 Mbyte sind, abzustürzen. In einem solchen Fall sind Sie auf den manuellen Transfer zum Server angewiesen. Für den Hausgebrauch sollte der Client allerdings ausreichen.

Fall 2: Windows soll, weil das System häufiger genutzt wird, als Printserver dienen

Für diese Einrichtung müssen Sie zuerst unter NetBSD Software nachinstallieren und zwar das Samba-Paket, das Sie der Aktualität und der Sicherheit halber aus dem Package-Tree nachinstallieren sollten. Eine Installation von der Installations-CD tut es aber auch, wenn Sie nur einen eingeschränkten Zugang zum Internet besitzen. Aus dem Samba-Paket selbst benötigen sie vor allem das Programm *smbclient* und einen *smbprint*-Druckfilter. Für letzteres können Sie eventuell den der Samba-Distribution beiliegenden Druckfilter verwenden, besser als die empirische Ermittlung der Konfiguration ist

allerdings ein anderes Konstrukt, das mit dem Magicfilter zusammenarbeitet, die Daten mundgerecht für die Windows-Maschine aufarbeitet und auf Anhieb funktioniert. Nach der Installation finden Sie das Programm *smbclient* unter »/usr/pkg/bin« und den mitgelieferten Druckfilter unter »/usr/pkg/share/examples/samba/printing«. Der Filter selbst befindet sich wie alle Binärprogramme aus den Packages ebenfalls in »/usr/pkg/bin«.

In diesem Fall ist der Windows-Drucker ein Tintenstrahler von HP. Für dieses Gerät wird die Konfigurationsdatei »printcap« nach folgendem Schema angepaßt:

```
lp0|hp500 auf windows98:\
:lp=/dev/lp0:\
:sd=/var/spool/lpd/lp0:\
        :sh:\
        :mx#0:
        :if=/usr/pkg/bin/smbprint:
        :lf=/var/log/lp-errs:
```

Tun Sie jetzt bitte folgende Dinge:

Legen Sie eine Logdatei an. Wenn Sie das Skript wie unten verwenden wollen, nehmen Sie das mit *touch /var/log/smb-print-log* vor. Der Eigentümer der Datei muß *root* und die Gruppe muß *wheel* sein.

Dann schreiben Sie das folgende Skript:

```
#!/bin/sh
logfile=/var/log/smb-print-log
/bin/date >> $logfile
/usr/pkg/libexec/magicfilter/dj500-filter| \
 smbclient \\\\Ihre_Windows_Maschine\\LP0 -U "Sie%ge_heim" \
 -c "translate; print -"
```

Der Parameter hinter *-U* ist der Username auf der Windows-Maschine, deren Namen anstelle des Platzhalters »Ihre_Windows_Maschine« anzugeben ist.

Fall 3: Man besitzt einen Drucker ohne eigenen Prozessor

Wenn Sie über Samba auf der Windows-Maschine drucken, sollte das Gerät funktionieren, vorausgesetzt, der Druckertreiber auf dem Windows-Rechner ist korrekt installiert.

Fall 4: Man will auf der NetBSD-Maschine drucken

Verkaufen Sie den GDI-Drucker bitte an einen überzeugten Windows-User und beschaffen Sie sich einen Drucker mit einem eigenen Prozessor. Der Arbeits- und Kostenaufwand für die zu beschaffende Windows-Software wird in keinem Verhältnis zum Wert des Druckers stehen.

13.2 Das Common Unix Printing System (CUPS)

von Dr. Bernd Kallenbach

Drucksysteme sind notwendiger und wichtiger Bestandteil eines jeden Betriebssystems. Sie ermöglichen den Ausdruck der unterschiedlichsten Dokumente auf lokalen Geräten und natürlich über das Netzwerk. Die globale Integration des Internets in innerbetriebliche Datennetze hat auch die Entwicklung von web-basierenden Anwendungen rasant vorangetrieben. Vor diesem Hintergrund und vor allem in Bezug auf bestehende Sicherheitsprobleme stellt sich die Frage, ob der weit verbreitete und verwendete Berkeley Line Printer Daemon »lpd« noch zeitgemäß ist, auch wenn er bei NetBSD bereits installiert ist und zum Grundlieferumfang des Betriebssystems gehört. Auch neuere Applikationen, wie LPRng oder PLP, ändern an den grundlegenden Fähigkeiten des *lpd* nichts. Als eine echte Alternative bietet sich hier CUPS [4] als plattformübergreifendes Drucksystem an.

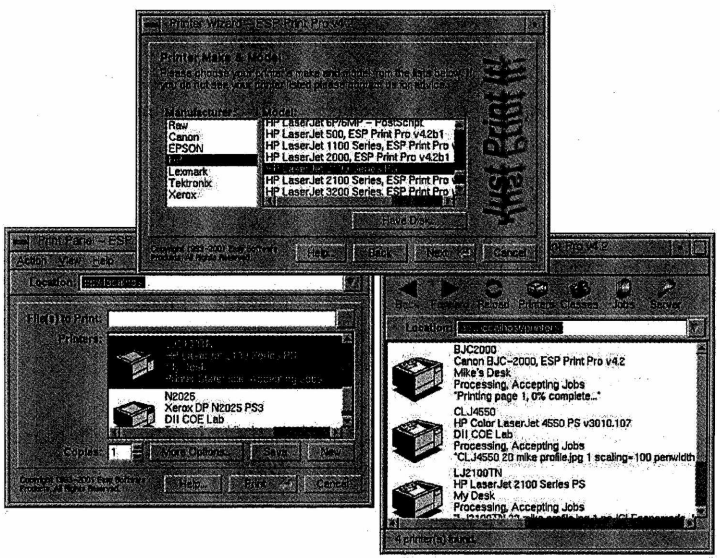


Abb. 13.1: Grafische Oberflächen zur Administration unter ESP Print Pro

CUPS hat in letzter Zeit nicht zuletzt wegen einer relativ unkomplizierten Verwaltung und Administration wesentlich an Bedeutung gewonnen. Aber weitaus wichtiger erscheint jedoch, daß es das erste Unix-Drucksystem ist, das das standardisierte *Internet Printing Protocol* (IPP) komplex umsetzt und hohen Sicherheitsansprüchen gerecht wird. Der IPP-Standard [1] umgeht die unterschiedlichen Schnittstellen und ermöglicht somit komfortabel systemübergreifende Druckvorgänge. Das »Common Unix Printing System« hat sich bereits kommerziell bewährt und rückt auch für den privaten Anwen-

der immer mehr in den Blickpunkt, zumal fast alle namhaften Drucker-Hersteller das IP-Protokoll unterstützen. CUPS unterstützt mit eingeschränkter Funktion die Protokolle des bekannten *lpd*, des Server Message Blocks (SMB) und darüber hinaus das HP-Printserver-Protokoll (JetDirect). CUPS ermöglicht die durchgängige netzwerkweite Druckerverwaltung (Network printer browsing) und unterstützt die PostScript Printer Description (PPD). In den PPD-Dateien sind alle notwendigen Informationen wie Schriftart, Farbe, Größe, Auflösung und andere erfaßt, die für ein ordnungsgemäßes Ausführen des Druckvorgangs notwendig sind. Wegen der einheitlichen Beschreibung sind diese Dateien plattformübergreifend verwendbar.

CUPS ist die Basis für das kommerzielle plattformübergreifende Drucksystem *ESP Print Pro* aus dem Hause Easy Software Products [6]. *ESP Print Pro* verarbeitet alle gängigen Formate, sichert qualitativ hochwertige Druckergebnisse und bedient mit über dreitausend Treibern fast die gesamte Druckerpalette. Für den kommerziellen Einsatz im Unternehmen ist *ESP Print Pro* also erste Wahl. Aktuell ist die Version 4.2.2. Für Interessenten kann eine voll funktionsfähige Demoversion (21 Tage) auf CD-ROM bestellt oder aus dem Internet bezogen werden.

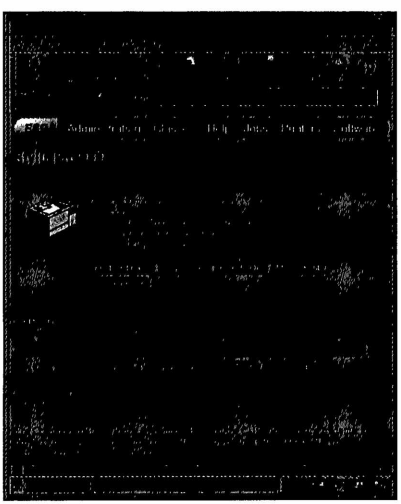


Abb. 13.2:
Web-Interface zum ESP-Drucksystem.

Durch umfassende grafische Administrationshilfen wird eine übersichtliche und einfache Druckerverwaltung sichergestellt. Alle Einstellungen können darüber hinaus über ein ebenfalls vorhandenes Web-Interface vorgenommen werden.

Aufgrund bereits genannter Vorzüge, einer überzeugenden Leistungsfähigkeit und nicht zuletzt der freien Verfügbarkeit bietet sich CUPS als primäres Drucksystem im kommerziellen und im privaten Umfeld an. Mittlerweile ist das Common Unix Printing System unter einer eine Vielzahl von Betriebssystemen und Distributionen lauffähig, darunter auch NetBSD.

Nachfolgend wird konkret auf die Installation und Administration unter NetBSD 1.6 eingegangen. Zum Verständnis der prinzipiellen Wirkungsweise werden einige allgemeine Ausführungen vorangestellt.

13.2.1 Kernel-lp-Device (Druckerport)

Es ist davon auszugehen, daß der Kernel eines installierten NetBSD die parallelen Ports (»/dev/lptx«, »/dev/lpax«) bereits unterstützt. Um das zu überprüfen, wird das System gestartet. Danach erfragen wir mit

```
# dmesg | grep lp
```

die Schnittstelle für den Drucker. Das Ergebnis wird dann etwa so aussehen:

```
lpt0 at isa0 port 0x378-0x37b irq 7
```

Jetzt ist es wichtig zu wissen, ob ein an diese Schnittstelle angeschlossenes Gerät generell (direkt) angesprochen werden kann. Das kann mit

```
# lptest 70 20 > /dev/lpt0
```

relativ einfach in Erfahrung gebracht werden, wobei alle druckbaren ASCII-Zeichen in zwanzig Zeilen mit jeweils siebzig Zeichen direkt an das Drucker-Device ausgegeben werden. Da in der Regel auf einem installierten NetBSD bereits ein Drucksystem (*http://i.e.lpd/*) aufgebracht ist, kann ein Druckauftrag in bekannter Art und Weise auch mit dem *lpr*-Befehl erfolgen:

```
# lpr -Pdrucker datei
```

13.2.2 Unix-Printing-System

Die allgemeine Darstellung in Abbildung 13.3 zeigt die Komponenten eines Unix-Drucksystems und verdeutlicht den Weg vom Auslösen eines Druckauftrags bis zur Übergabe der Daten an den Drucker selbst. Dieser Druckauftrag (Job) kann von einem beliebigen Anwender unter Angabe des Dokumentennamens und ergänzender Optionen wie Formatierung und Auflösung ausgelöst werden.

Die Daten eines Druckauftrags liegen in der Regel im Postscript-Format vor (es gibt Ausnahmen!). Postscript ist, vereinfacht ausgedrückt, eine standardisierte Beschreibungssprache für Druckseiten, die im professionellen Bereich fast ausschließlich verwendet wird und viele Vorteile bietet. Postscript-Drucker, die derartige Druckaufträge *direkt* ausführen können, sind relativ teuer und vorzugsweise für den kommerziellen Einsatz konzipiert. Für private und semiprofessionelle Anwendungen hat sich deshalb der Einsatz eines sogenannten Postscript-Interpreters, der die Steuerinformationen in druckerspezifische Daten umsetzt, bestens bewährt und durchgesetzt. Ein solcher frei

verfügbarer Interpreter ist *Ghostscript*. Dieses Programm unterstützt eine Vielzahl von Druckern der verschiedensten Hersteller. Die Liste der unterstützten Geräte wird ständig erweitert und verbessert.

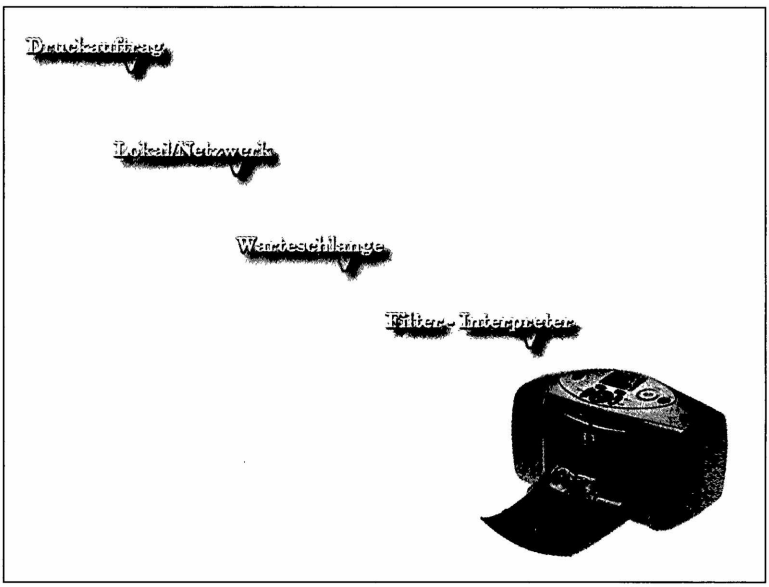


Abb. 13.3: Vereinfachte Darstellung eines Drucksystems unter Unix

Wer den Neukauf eines Druckers plant, sollte vorab einen Blick auf die Liste der von Ghostscript unterstützten Drucker werfen, um Fehlkäufe zu vermeiden. Wie wir später noch sehen werden, ist unter bestimmten Voraussetzungen eine Verwendung nicht aufgeführter Drucker durch Integration neuer Treiber möglich.

13.2.3 Drucker-Spooling

Eine zentrale Stellung nimmt das sogenannte Drucker-Spooling ein. Was ist darunter zu verstehen?

Ein Drucker-Spooling-System unter Unix besteht im allgemeinen aus einem Programmpaket, das folgende Aufgaben zu erfüllen hat:

- Kopieren der Druckdaten lokal oder über das Netzwerk in das Spooling-Verzeichnis des gewählten Druckers, zum Beispiel »/var/spool/output/lpd/«.
- Einordnen der Jobs in die Warteschlange (Queue) des ausgewählten Druckers.
- Warten auf die Druckerfreigabe bei mehreren Druckaufträgen.

- Weiterleitung der Postscript-Daten an den Interpreter (also Ghostscript) und Ausdruck der druckerspezifisch aufbereiteteten Informationen.
- Säubern des Spooling-Verzeichnisses nach Beendigung des Druckauftrags und Auslösen von Meldungen bei fehlerhafter Job-Ausführung.

Die noch häufig eingesetzte Spooling-Software ist der *lpd* (Line-Printer-Daemon). Dieser Name steht sowohl für den eigentlichen Dämonen als auch für das komplette Spooling-System!

Dieses Softwarepaket baut auf veralteten Strukturen auf und enthält erhebliche Sicherheitslücken. Also ein guter Grund, sich für CUPS als primäres Drucksystem zu entscheiden.

Für das Spooling selbst ist eine Reihe von Programmen erforderlich:

- *lpd*: Der Line-Printer-Daemon selbst. Er sorgt für die Ausgabe der Daten auf den jeweiligen Drucker. Der Dämon wird in der Regel beim Systemstart über ein entsprechendes Skript gestartet. Um festzustellen, welche Druckerwarteschlangen (Queues) zu verwalten sind, wird die Datei »/etc/printcap« abgefragt. Liegt ein Druckauftrag vor und ist der bezeichnete Drucker bereit, leitet der *lpd* die Daten zum Ausdruck weiter.
- *lpr*: Es werden durch den Anwender Druckaufträge ausgelöst, die an den *lpd* weitergeleitet werden. Beispielsweise gibt *lpr -Plaserjet test.ps* die Datei »test.ps« über den Drucker *laserjet* aus.
- *lpq*: Listet die Druckeraufträge der Warteschlange (Queue) eines benannten Drukkers auf.
- *lprm*: Entfernt Aufträge aus einer Warteschlange (Queue).

Informationen zum Gebrauch der einzelnen Befehle können über die Manpages erfragt werden.

13.2.4 Filter

Die meisten Unix-Programme geben ihre die Dokumente bereits im Postscript-Format aus. Oft ist es jedoch erforderlich, die unterschiedlichsten Datentypen aufzubereiten und in druckerverständliche Informationen umzuwandeln. Diese Aufgabe übernehmen sogenannte »Magicfilter«. Einer der bekanntesten und besten Filter dieser Art ist *apsfilter* [3]. Er ist Bestandteil jeder guten Unix-Distribution und kommt mit allen denkbaren Datentypen klar. *apsfilter* verfügt darüber hinaus über ein eigenes Setup-Programm für die »printcap«-Datei.

13.2.5 Ghostscript

Eines der wichtigsten Programme für den Druck unter Unix ist *Ghostscript* [6], ein freier Interpreter, der Postscript-Befehle direkt in eine druckerverständliche Form umwandelt. Damit ist es möglich, preisgünstige Drucker, die nicht über diese Funktionalität verfügen, entsprechend zu erweitern.

Anmerkung

Aus der Kombination eines *Magicfilters* wie *apsfilter* und dem freien Interpreter *Ghostscript* entsteht ein preiswerter *virtueller Postscript-Drucker*, der fast alle denkbaren Datentypen verarbeiten kann.

Für die meisten privaten Anwendungsfälle ist das die Lösung der Wahl, zumal für die gängigsten Druckertypen entsprechende Treiber zur Verfügung stehen.

CUPS selbst verwendet eine speziell angepaßte Version des GNU-Ghostscript-Interpreters, das sogenannte ESP Ghostscript in der aktuellen Version 7.05.5. Die Pakkage-Collection von NetBSD 1.6 enthält bereits die für NetBSD aufbereitete Ghostscript-Version, so daß der Installationsprozeß einfach mit

```
# cd /usr/pkgsrc/print/ghostscript-eps
# make install
```

durchgeführt werden kann. Das Quellarchiv wird dabei automatisch beschafft und die Quellen werden extrahiert. Danach erfolgt das Hinzufügen der erforderlichen Patches zum Kompilieren unter NetBSD und das Erstellen des Pakets selbst. Fehlende Quellen werden automatisch besorgt, übersetzt und in das System integriert. Abschließend erfolgt die Installation des Ghostscript-Interpreters selbst. Diese Vorgehensweise setzt natürlich eine funktionierende Internetverbindung voraus.

Da die Installation protokolliert wird, kann der Erfolg mit

```
# pkg_info -a
```

überprüft werden. In der Liste der installierten Programme muß jetzt der Eintrag *ghostscript-esp-7.05.5* (EPS/CUPS Postscript Interpreter) erscheinen. Danach schaffen wir mit

```
# make clean-depends && make clean
```

wieder etwas Platz. Als nächstes geben wir auf der Kommandozeile

```
#gs --help
```

ein und erfragen die Funktionalitäten und die integrierten Treiber des installierten Ghostscript-Interpreters (Listing 13.1).

```
ESP Ghostscript 7.05.5 (2002-09-02)
Copyright (C) 2002 artofcode LLC, Benicia, CA.  All rights reserved.
Usage: gs [switches] [file1.ps file2.ps ...]
Most frequently used switches: (you can use # in place of =)
 -dNOPAUSE           no pause after page   | -q       `quiet', fewer messages
 -g<width>x<height>  page size in pixels   | -r<res>  pixels/inch resolution
 -sDEVICE=<devname>  select device         | -dBATCH  exit after last file
 -sOutputFile=<file> select output file: - for stdout, |command for pipe,
                                       embed %d or %ld for page #
Input formats: PostScript PostScriptLevel1 PostScriptLevel2 PDF
Available devices:
   x11 x11alpha x11cmyk x11mono x11cmyk2 x11cmyk4 x11cmyk8 x11rg16x x11rg32x
   x11gray2 x11gray4 bbox bff bit bitcmyk bitrgb bmp16 bmp16m bmp256 bmp32b
   bmpa16 bmpa16m bmpa256 bmpa32b bmpamono bmpasep1 bmpasep8 bmpgray bmpmono
   bmpsep1 bmpsep8 ccr cfax cgm24 cgm8 cgmmono cif dfaxhigh dfaxlow epswrite
   faxg3 faxg32d faxg4 inferno jpeg jpeggray mag16 mag256 mgr4 mgr8 mgrgray2
   mgrgray4 mgrgray8 mgrmono miff24 pbm pbmraw pcx16 pcx24b pcx256 pcx2up
   pcxcmyk pcxgray pcxmono pdfwrite pgm pgmraw pgnm pgnmraw pkm pkmraw pksm
   pksmraw plan9bm png16 png16m png256 pnggray pngmono pnm pnmraw ppm ppmraw
   psgray psmono psrgb pswrite sgirgb sunhmono sxlcrt tiff12nc tiff24nc
   tiffcrle tiffg3 tiffg32d tiffg4 tifflzw tiffpack alc2000 alc4000 alc8500
   alc8600 ap3250 appledmp atx23 atx24 atx38 bj10e bj10v bj10vh bj200 bjc600
   bjc800 bjc880j bjccmyk bjccolor bjcgray bjcmono cdeskjet cdj1600 cdj500
   cdj550 cdj670 cdj850 cdj880 cdj890 cdj970 cdjcolor cdjmono chp2200 cljet5
   cljet5c cljet5pr coslw2p coslwxl cp50 declj250 deskjet dj505j djet500
   djet500c djet820c dl2100 dmprt dnj650c epl2050 epl2050p epl2120 epl5800
   epl5900 epl6100 eps9high eps9mid epson epsonc escp escpage escpc fmlbp
   fmpr fs600 gdi hl1240 hl1250 hl7x0 hpdj1120c hpdj310 hpdj320 hpdj340
   hpdj400 hpdj500 hpdj500c hpdj510 hpdj520 hpdj540 hpdj550c hpdj560c
   hpdj600 hpdj660c hpdj670c hpdj680c hpdj690c hpdj850c hpdj855c hpdj870c
   hpdj890c hpdjplus hpdjportable ibmpro imagen iwhi iwlo iwlq jetp3852
   jj100 la50 la70 la75 la75plus laserjet lbp310 lbp320 lbp8 lex2050 lex3200
   lex6700 lex7000 lips2p lips3 lips4 lips4v lj250 lj3100sw lj4dith lj4dithp
   lj5gray lj5mono ljet2p ljet3 ljet3d ljet4 ljet4d ljet4pjl ljetplus ln03
   lp1800 lp1900 lp2000 lp2200 lp2563 lp3000c lp7700 lp8000 lp8000c lp8100
   lp8200c lp8300c lp8300f lp8400f lp8500c lp8600 lp8600f lp8700 lp8800c
   lp8900 lp9300 lp9400 lp9600 lp9600s lq850 lx5000 lxm3200 lxm5700m m8510
   md1xMono md2k md50Eco md50Mono md5k mj500c mj6000c mj700v2c mj8000c ml600
   necp6 npdl oce9050 oki182 oki4w okiibm paintjet pcl3 photoex picty180 pj
   pjetxl pjxl pjxl300 pr1000 pr1000_4 pr150 pr201 pxlcolor pxlmono r4081
   samsunggdi sj48 st800 stcolor t4693d2 t4693d4 t4693d8 tek4696 uniprint
   xes cups ijs stp nullpage
Search path:
   . : /usr/pkg/share/ghostscript/7.05/lib :
   /usr/pkg/share/ghostscript/fonts :
   /usr/pkg/share/fonts/default/ghostscript :
```

```
    /usr/pkg/share/fonts/default/Type1 :
    /usr/pkg/share/fonts/default/TrueType : /usr/lib/DPS/outline/base :
    /usr/openwin/lib/X11/fonts/Type1 : /usr/openwin/lib/X11/fonts/TrueType
For more information, see /usr/pkg/share/doc/html/ghostscript/Use.htm.
Report bugs to bug-gs@ghostscript.com, using the form in Bug-form.htm.
```

Listing 13.1: Kommandozeilenausgabe von ESP Ghostscript

Damit sind alle wichtigen und dem Verständnis dienenden Details zur Realisierung von Druckdiensten unter Unix beschrieben.

13.2.6 Installation und Administration von CUPS

Um das Common Unix Printing System zu installieren und fehlerfrei anzuwenden, ist es notwendig, das alte Drucksystem außer Betrieb zu nehmen. CUPS bringt alle Funktionalitäten selbst mit. Alle bekannten Befehle des Berkeley *lpd* (*lpr, lpc, lpq, lprm*) sind weiterhin anwendbar. Auch System-V-Anweisungen wie *lp, accept, reject, cancel, enable, disable, lpinfo, lpadmin, lpmove* und *lpstat* besitzen ihre Gültigkeit. Unterschiede in der Anwendung sind in den entsprechenden Manpages dokumentiert.

Damit der alte Druckdämon beim Systemstart nicht aktiviert wird, ist der Eintrag in der »rc.conf« zu ändern, und zwar von *lpd=YES* nach *lpd=NO*. Das alte Drucksystem kann natürlich auch entfernt werden. Eine weitere Möglichkeit ist, wie unten gezeigt, die Umbenennung der relevanten Dateien. Darüber hinaus empfiehlt es sich, eine bereits vorhandene »printcap«-Datei zu sichern. Das kann mit

```
# cp /etc/printcap /etc/printcap.save
```

erfolgen. CUPS liegt in der aktuellen Version 1.1.16 vor. Das Aufbringen des Printing-Systems erfolgt wie bereits zu Ghostscript beschrieben über die Package-Collection mit

```
# cd /usr/pkgsrc/print/cups
# make install.
```

Alle notwendigen Bibliotheken und Utilities werden installiert. Genaue Auskunft gibt wieder *pkg_info -a*. Nach Abschluß des Installationsprozesses sind einige Eintragungen und Systemanpassungen zwingend erforderlich, um das Funktionieren zu gewährleisten. Dazu müssen wir wissen, daß sich nach der Installation die CUPS-relevanten Dateien im Verzeichnis »/usr/pkg/bin« befinden. Um nicht mit den gegebenenfalls vorhandenen Dateien des alten Drucksystems zu kollidieren, wird sicherheitshalber umbenannt:

```
# mv /usr/bin/lp /usr/bin/lp_bsd
#mv /usr/bin/lpr /usr/bin/lpr_bsd
#mv /usr/bin/lpq /usr/bin/lpq_bsd
#mv /usr/bin/lprm /usr/bin/lprm_bsd
```

Danach ist, falls noch nicht geschehen, das Spooling-Verzeichnis anzulegen:

```
# mkdir /var/spool/output
#mkdir /var/spool/output/lpd
```

Um bei jedem Systemstart den CUPS-Server zu starten, erfolgt mit

```
# echo "cupsd=YES" >> /etc/rc.conf
```

der notwendige Eintrag.

Damit das Startskript *cupsd* ordnungsgemäß ausgeführt werden kann, muß es umkopiert werden:

```
# cp  /usr/pkg/etc/rc.d/cupsd /etc/rc.d
```

Danach ist das System bei eingeschaltetem Drucker neu zu starten. Mit

```
# ps -ax | grep cupsd
188 ?? Ss              0:00.69  /usr/pkg/sbin/cupsd     #Beispiel
```

kann dieser Prozeß dokumentiert werden.

CUPS stellt unter anderem folgende Programme zur Verfügung, die auf der Kommandozeile ausgeführt werden können:

- *lpr* zum Ausdruck von Dokumenten auf einem gewählten Drucker.
- *lpstat*. Dieses Programm zeigt die CUPS-Statusinformationen an.
- *lpadmin* zur Administration von Druckern.
- Mit *cancel* werden Druckeraufträge beendet.
- *accept* und *reject* dienen der Annahme und Zurückweisung von Druckaufträgen am gewählten Drucker.
- *enable* und *disable* aktivieren und deaktivieren den bezeichneten Drucker.

Damit sind alle Vorbereitungen zur Administration von CUPS getroffen und in einem weiteren Schritt ist ein erster Drucker zu installieren. Diese Aufgabe kann einmal über die Kommandozeile erfolgen und, wie zu erwarten, über ein Web-Interface. Hier werden beide Möglichkeiten aufgezeigt.

13.2.7 Administration

Die Administration über die Kommandozeile erfolgt als User *root* mit dem Befehl *lpadmin -p DRUCKERNAME -m DRUCKER.PPD -v DEVICE -E*. Die Parameter bedeuten:

DRUCKERNAME	Der Name, unter dem der Drucker unter CUPS bekannt gemacht wird.
DRUCKER.PPD	Die Datei für den auswählten Drucker.
DEVICE	Das Gerät beziehungsweise Device, an das der Drucker angeschlossen ist.

Hier eine Liste der wichtigsten Optionen:

-c Klasse	Fügt den bezeichneten Drucker in eine Klasse ein.
-i Interface	Setzt ein System V-Skript für den bezeichneten Drucker.
-m Model	Setzt einen Treiber aus der Treiber-Bibliothek.
-o name=Werte	Setzt Druckeroptionen, die mit dem Befehl *lpoptions -l* aufgelistet werden können.
-r Klasse	Entfernt den bezeichneten Drucker aus einer Klasse.
-D Text	Beschreibung des Druckers.
-E	Aktiviert den bezeichneten Drucker.
-P ppd-Datei	Überschreibt die Option *-i* und legt die zu verwendende PPD-Datei fest.
-v Gerät	Bezeichnet die Schnittstelle für den Drucker.

Und hier noch das Beispiel einer praktisch verwendbaren Installation:

```
# lpadmin -p Foto_lokal -m stphoto.ppd  -v parallel:/dev/lpt0 -E
```

Ein Drucker der Serie Epson-Stylus-Color (»stphoto.ppd«) wird unter dem Namen *Foto_lokal* (das ist der Name der Warteschlange) für den Standard-Druckerport »/dev/lpt0« auf dem lokalen Computer installiert und aktiviert (*-E*).

Es wird an dieser Stelle darauf verwiesen, daß entsprechend des IP-Protokolls die Bezeichnungen der Drucker nicht casesensitiv sind! Es wird also nicht zwischen Groß- und Kleinschreibung unterschieden. Damit bezeichnen die Ausdrücke *Foto_lokal* und *foto_lokal* den gleichen Drucker!

13.2.8 Drucker-Klassen in CUPS

CUPS gestattet es, Drucker in Klassen einzuordnen, die funktionellen oder thematischen Gruppen zugeordnet werden können. Die Zuordnung des benannten Druckers wird über die Option *c* im *lpadmin*-Befehl erreicht.

So können zum Beispiel alle fotofähigen Drucker einer Klasse *Foto* und alle Laserdrucker einer Klasse *Laser* zugeordnet werden. Es ist auch denkbar, verschiedene Drucker in einer Klasse *Verwaltung* thematisch zu organisieren. Ein Druckvorgang zur einer definierten Klasse spricht immer den ersten der verfügbaren Drucker an. Damit wird ein automatisches Load Balancing erreicht.

```
# lpadmin -p Foto_lokal -m stphoto.ppd -v parallel:/dev/lpt0 -c buero -E
```

Der Drucker *Foto_local* wird angelegt und der Klasse *buero* zugeordnet. Ist die Klasse nicht vorhanden, wird sie neu erzeugt. Der Befehl

```
# lpr -C buero datei
```

löst dann einen Druckvorgang zur definierten Klasse aus. Durch eine geschickte Klassenwahl läßt sich eine außerordentlich hohe Ausfallsicherheit, insbesondere bei großen Netzen, erreichen.

13.2.9 Druckerinstanzen in CUPS

Ein weiteres Feature von CUPS ist die Verwendung von Druckerinstanzen. Was ist darunter zu verstehen? Eine Druckerinstanz ist eine Sammlung definierter Optionen für einen gewählten Drucker. Diesem Drucker können verschiedene Merkmale zugeordnet werden. Es ist möglich, einem Drucker mehrere Instanzen für unterschiedliche Aufgaben zuzuweisen, die jedem Benutzer zur Verfügung stehen. Ein Beispiel:

Ein Drucker *laserjet* soll zwei Seiten auf einem Blatt ausdrucken. Wir weisen ihm die Instanz *draft* zu. Und benutzen dazu:

```
# lpoptions -p laserjet/draft -o number-up=2
```

Nun kann für den Druckvorgang eines Blattes mit zwei Seiten einfach die Instanz aufgerufen werden:

```
# lpr -Plaserjet/draft datei
```

Mit

```
# lpoptions -p laserjet/draft
number-up=2                          # Ergebnis der Abfrage
```

wird die Option bei Ausführung der Instanz erfragt.

Instanzen können beliebig angelegt und mit

```
# lpoptions -x drucker/instanz
```

gelöscht werden. Aus den bisherigen Ausführungen ist bereits zu erkennen, welche mächtigen Werkzeuge CUPS dem Administrator in die Hand gibt.

13.2.10 Netzwerkdruck mit CUPS

CUPS unterstützt mehrere Methoden zum Druck über ein Netzwerk. Neben der Ausführung von Druckaufträgen über andere CUPS-Server können auch andere Systeme (Windows, Macintosh) einbezogen werden. Dabei werden die entsprechenden Drucker automatisch verwaltet, wenn zum Beispiel mehrere (CUPS-) Server im gleichen Subnetz aktiv sind (Printer Sharing).

Im folgenden Beispiel wird die Druckerwarteschlange (*lp*) eines lpd-organisierten entfernten Rechners (192.168.110.1) über das Netzwerk unter den Namen *Laser_remote* ohne Aktivierung der eigenen Warteschlange angesprochen:

```
# lpadmin -p Laser_remote -m laserjet.ppd -v lpd:// 192.168.110.1/lp
```

Auch in größeren Netzen sichert CUPS die komplexe Administration der Druckerdienste in einfachster Art und Weise. Es muß an dieser Stelle darauf hingewiesen werden, daß die Verwaltung von Remote-Druckern mit diesem Interface nur bedingt möglich ist, da der CUPS-Server selbst auf dem lokalen Rechner installiert ist. Die Kontrolle administrativer Aufgaben übernimmt der *Security option block* der CUPS-Konfigurationsdatei. Es ist denkbar, darin die Adresse des Localhost gegen die Subnetznummer des verwendeten Netzwerkes auszutauschen, um damit den Zugriff auf alle Netzwerkdrucker zu erreichen. Darüber hinaus gibt es eine Vielzahl nutzbarer Möglichkeiten, textbezogen oder grafisch orientiert administrative Aufgaben innerhalb des Netzwerks zu organisieren. An dieser Stelle wird nicht gesondert darauf eingegangen.

13.2.11 Das Web-Interface

Weitaus ansprechender und komfortabler als über die Kommandozeile kann CUPS über ein Web-Interface verwaltet werden. Der CUPS-Server reserviert dafür den Port 631, was den IPP-Richtlinien entspricht.

Als beliebiger Nutzer wird nun ein Browser gestartet. Der genannte Port 631 wird über die Adresse *http://localhost:631/* angesprochen.

Das Web-Interface ist mit Abb. 13.2 identisch. Aus der Kopfleiste kann die gewünschte Option zu Bearbeitung ausgewählt werden. Über den Auswahlpunkt *Hilfe* ist der Zugriff auf eine umfangreiche Dokumentensammlung möglich.

Bei der Erstinstallation eines Druckers ist *Administration* zu wählen und nach Eingabe von User-ID (*root*) und Paßwort (in diesem Fall des *root*-Paßworts) kann mit der Druckerinstallation begonnen werden. Dabei sind in den entsprechenden Feldern die erfor-

derlichen Eingaben entsprechend den Argumenten der Kommandozeile in der oben beschriebenen Form vorzunehmen. Alle weiteren Angaben sind selbstdokumentierend und dürften keine Probleme bereiten.

Jeder der installierten Drucker kann einzeln beliebig angepaßt und konfiguriert werden, indem der entsprechende Link zum Drucker aufgerufen wird. Auch die Verwaltung der druckerbezogenen Jobs ist jederzeit und ohne großen Aufwand möglich.

Nach erfolgreicher Übernahme der Drucker in CUPS ist ein Testausdruck zur Beurteilung der gewählten Treiber sehr empfehlenswert. Nach dem erfolgreichen Abschluß dieser Arbeiten ist CUPS als primäres Drucksystem für eine Workstation eingerichtet. Steht für Serveranwendungen kein grafisches System zur Verfügung, kann eine webbezogene Verwaltung unter Heranziehung des bekannten Lynx-Browsers erfolgen.

Für den täglichen Umgang mit CUPS stehen neben dem Webinterface auch grafisch orientierte Hilfsprogramme zur Verfügung.

13.2.12 Administrative Grafiktools

X Printing Panel

Das X Printing Panel XPP [7] ist frei verfügbar und basiert auf dem FL-Toolkit. Es ist in der Version 1.1 Bestandteil der NetBSD Package Collection. Der Ablauf der Installation von Programmen aus der Package Collection wurde an anderer Stelle beschrieben.

Das X Printing Panel befindet sich unter »/usr/pkgsrc/print/xpp«. Aufgerufen wird es mit *xpp*.

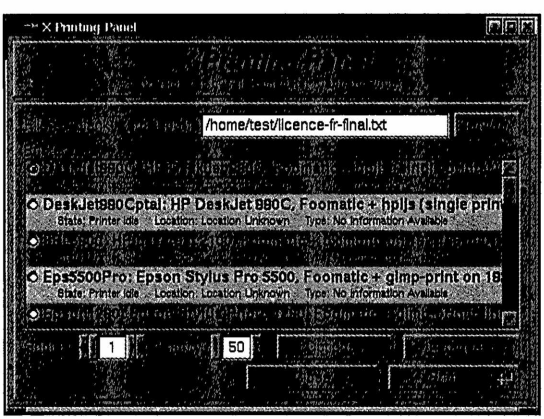


Abb. 13.4:
Das X Printing Panel

Die Einstellmöglichkeiten von *xpp* sind vielfältig. Neben den notwendigen Grundeinstellungen werden auch farb-, text- und fotospezifische Einstellungen angeboten, die über den Menüpunkt *Optionen* erreichbar sind. Über einen Browser können die zum Druck vorgesehenen Dokumente ausgewählt werden. XPP ist ein einfach zu installie-

rendes und gut verwendbares Hilfsmittel. Als Icon auf dem Desktop abgelegt, steht es jederzeit zur Verfügung.

Anwendung von PPD-Dateien

Wie oben angesprochen steht PPD für Postscript Printer Description. Die Syntax einer PPD wurde vom Unternehmen Adobe genau spezifiziert. Die PPD-Dateien sind meist druckerspezifisch und legen fest, welche Eigenschaften der Drucker aufweisen soll:

- Papierformate und Rasterauflösungen,
- Schrift-Arten, Druckspezifika (beispielsweise doppelseitig),
- Endverarbeitung (beispielsweise Heften) und die
- Befehlsfolge für den korrekten Ausdruck.

PPDs können auch Anweisungen definieren, die keine Standard-Postscript-Befehle sind. Damit haben die Hersteller die Möglichkeit, ihre Drucker komplex und trotzdem plattformübergreifend zu beschreiben. CUPS kann fast uneingeschränkt diese PPDs auswerten. Ausnahme davon bilden nur wenige High-End-Drucker, die zusätzlich noch DLLs (Dynamic Link Libraries) verwenden, die vom Druckertreiber aus aufgerufen werden, um bestimmte Funktionalitäten bereitzustellen.

Bei der CUPS-Entwicklung wurde die Beschreibung von Druckerfeatures und ihre Ansteuerung auch für Nicht-Postscript-Drucker in eine PPD-Syntax gepackt. Somit kann der Druckerhersteller seine eigenen Treiber über eine PPD in das CUPS-System einbinden. Durch die Schaffung dieser einheitlichen Schnittstelle ist die teure Entwicklung zusätzlicher Unix-Treiber für Drucker-Produkte nicht mehr erforderlich!

CUPS hat nach der Installation nur eine begrenzte Zahl von Treibern zur Verfügung. Dieser Pool kann mit einer Vielzahl frei verfügbarer PPDs erweitert werden. Hierfür ist unter *http://www.linuxprinting.org/* eine Datenbank frei verfügbar, aus der sich CUPS-PPDs fast aller Nicht-Postscript-Drucker generieren lassen. Dazu gibt man den Drukker an und wählt einen zugehörigen Ghostscript-Filter aus. Auf der Web-Site wird mit dem CUPS-O-Matic-Script eine PPD erstellt. Um die so erzeugte Datei verwenden zu können, ist das Filterscript *cupsomatic* auf dem lokalen Rechner erforderlich. Dabei handelt es sich um ein Perlskript, das von CUPS zur Interpretation der PPD-Dateien aufgerufen wird. Zur ordnungsgemäßen Funktion muß dieses Skript ausführbar und lesbar sein. Installiert wird es in bekannter Art mit

```
# cd /usr/pkgsrc/print/cupsomatic
# make install
# make clean
# make clean-depends
```

Abschließend kann gesagt werden, daß CUPS heute wohl mehr Drucker unterstützt als jedes andere Drucksystem.

Literatur

[1] Grant Taylor: The Linux Printing HowTo, *http://www.picante.com/~gtaylor/ pht/*.

[2] GNU/Linux Printing: The Printing HOWTO, *http://www.linuxprinting.org/*.

[3] GNU a2ps: Any to PostScript Filter, *http://www.inf.enst.fr/~demaille/a2ps/*.

[4] CUPS: Common Unix Printing System, *http://www.cups.org/*.

[5] IPP: The Internet Printing Protocol, *http://www.ieft.org/*.

[6] ESP:ESP Print Pro, *http://www.easysw.com/*.

[7] XPP: X Printing Panel, *http://cups.sourceforge.net/xpp*.

14 Das Init-Konzept von NetBSD

von Wolfgang Soltendick

Nachdem der Kernel geladen wurde und die Hardware initialisiert hat, wird bei Unix-Systemen das »init«-Programm aufgerufen. Dazu wird die Datei »/etc/rc« ausgeführt, wobei ihr als erstes Argument das Wort *autoboot* mit übergeben wird, sofern vorher nicht der Einzelbenutzermodus aktiv war. Liefert »/etc/rc« einen Wert zurück, der ungleich 0 ist, wird davon ausgegangen, daß beim Starten ein Fehler aufgetreten ist. In diesem Fall wird der Einzelbenutzermodus gestartet, so daß der Administrator Gelegenheit hat, die Probleme zu beseitigen.

Ist der Rückgabewert von »/etc/rc« dagegen 0, werden als nächstes die Terminalemulationen und die Konsole gestartet, die dann vom *init*-Prozeß weiter unterhalten werden. *init* liest und interpretiert die Datei »/etc/ttys« und führt das Kommando aus, das in einer Konfigurationszeile in der zweiten Spalte angegeben ist. Dabei handelt es sich in der Regel um das Programm *getty*, das die entsprechende Leitung öffnet und initialisiert. Außerdem führt es das *login*-Programm für diesen Port aus. Durch *login* wird eine Shell gestartet, wenn ein gültiger Benutzer sich angemeldet hat. Wird diese Shell beendet, weil beispielsweise der Benutzer sich abmeldet oder sie zwangsweise beendet wird, löscht das *init*-Programm den Benutzer aus der Datei »/var/run/utmp« und protokolliert den Vorgang in der Datei »/var/log/wtmp«. Anschließend wird ein neuer *getty*-Prozeß für diesen Port gestartet. Ist das Starten der *getty*-Prozesse erfolgreich, steht das System den Benutzern zur Verfügung und zwar solange, bis der *init*-Prozeß ein bestimmtes Signal empfängt.

Der Status eines Terminal-Ports wird in der Textdatei »/etc/ttys« geändert, ohne daß dazu der Rechner neu gestartet werden müßte. Das erfolgt durch Senden eines *SIGHUP*-Signals an den *init*-Prozeß mit dem Kommando

```
# kill -s HUP 1
```

Dabei bezeichnet die 1 die Nummer des Prozesses, der das Signal empfangen soll. Der *init*-Prozeß besitzt immer die Nummer 1, da er vom Kernel zuerst gestartet wurde. Nachdem *init* das Signal empfangen hat, wird die Datei »/etc/ttys« erneut gelesen. Wird ein Terminal in dieser Datei abgeschaltet, sendet *init* ein *SIGHUP*-Signal an den zugehörigen Kontrollprozeß. Bei allen Leitungen, die bisher abgeschaltet waren und nun angeschaltet sind, wird der zugehörige *getty*-Prozeß gestartet, um ein Login zu ermöglichen. Wird der Status eines Terminals während der Sitzung geändert, wird diese Änderung erst dann wirksam, wenn der Benutzer sich abgemeldet hat und *init* einen neuen *getty*-Prozeß für dieses Terminal startet. Wenn eine Zeile auskommentiert oder gelöscht wird, passiert nichts. Allerdings wird dadurch die Synchronisation zwischen den Terminals und den Einträgen in der Datei »/var/run/utmp« gestört, so daß dieses Vorgehen nicht angewendet werden sollte. Konnten die Aktionen nach dem Auftreten des Signals *SIGHUP* erfolgreich ausgeführt werden, wird das System wieder in den Mehrbenutzermodus versetzt. Dann kann mit ihm weitergearbeitet werden.

Empfängt der *init*-Prozeß ein Terminal-Stop-Signal (*SIGTSTP*), beispielsweise über

```
# kill -s TSTP 1
```

wird in den sogenannten Boring-Modus geschaltet. In ihm können keine neuen Sitzungen (*getty*-Prozesse) mehr gestartet werden, so daß das System so langsam beendet werden kann. Wird in diesem Modus ein *SIGHUP*-Signal an den *init*-Prozeß gesendet, kehrt er wieder in den normalen Mehrbenutzermodus zurück. Alternativ startet ein *SIGTERM* den Einzelbenutzermodus. Ansonsten sind in diesem Modus die Signale gültig, die hier für den Mehrbenutzermodus beschrieben werden, mit eben diesen Auswirkungen. Ein weiteres Signal ist *SIGTERM*. Damit wird das System in den Shutdown-Modus versetzt. In diesem Fall werden die Mehrbenutzer-Operationen beendet. Das erfolgt beispielsweise mit

```
# kill -s TERM 1
```

Danach wird der Einzelbenutzermodus gestartet. Wenn Prozesse vorhanden sind, bei denen ein Deadlock aufgetreten ist, wartet *init* nicht solange, bis sie tatsächlich beendet sind. Stattdessen tritt nach dreißig Sekunden ein Timeout auf und man sieht eine entsprechende Warnmeldung auf dem Bildschirm.

Für das korrekte Funktionieren des NetBSD-Systems ist die problemlose Ausführung des *init*-Prozesses entscheidend. Aus diesem Grund wird auch das System neu gestartet, wenn er beendet wird. Kann *init* beim Hochfahren nicht lokalisiert werden, stoppt das System mit einer *panic*-Meldung. In diesem Fall geht gar nichts mehr und man kann mit einem Rettungssystem wie der Buch-CD versuchen, die Ursache dafür herauszufinden.

14.1 Die »/etc/rc«

Eine zentrale Rolle beim Starten von NetBSD spielt die Datei »/etc/rc«. Mit Hilfe dieses Skripts wird beispielsweise gesteuert, welche Variablen global gesetzt werden. Wurde vorher nicht der Einzelbenutzermodus gestartet, wird der Datei als Parameter der String *autoboot* übergeben. Dadurch wird die zugehörige Variable (*autoboot*) auf den Wert *YES* gesetzt.

Außerdem werden aus der Datei »/etc/rc.subr« einige Shell-Funktionen geladen, die später zum Einsatz kommen. Danach wird die Datei »/etc/rc.conf« gelesen. In ihr ist festgelegt, welche Dienste beim Hochfahren des Systems gestartet werden. Da es sich um ein Skript handelt, muß es gültige Shell-Kommandos enthalten. Zusätzlich werden in der »rc.conf«-Datei Variablen gesetzt, die entweder angeben, ob ein Dienst gestartet werden soll oder welche Parameter dem entsprechenden Dienst mit übergeben werden sollen. Im ersten Fall kann die Variable nur den Wert *YES* oder *NO* angeben (Groß- und Kleinschreibung spielen keine Rolle, *yes* und *no* sind damit auch gültig), ansonsten sind die entsprechenden Argumente zu spezifizieren. Erwartet die Datei »/etc/rc«, daß eine bestimmte Variable gesetzt ist oder besitzt sie einen Wert, der nicht erlaubt ist, wird eine entsprechende Warnung ausgegeben.

```
#!/bin/sh
#
# $NetBSD: rc,v 1.152.4.1 2000/08/23 12:05:17 lukem Exp $
#
# rc.sh --
#       Run the scripts in /etc/rc.d with rcorder.
#       System startup script run by init on autoboot or after single-user.
#       Output and error are redirected to console by init, and the console
#       is the controlling terminal.

export HOME=/
export PATH=/sbin:/bin:/usr/sbin:/usr/bin

. /etc/rc.subr
. /etc/rc.conf

if ! checkyesno rc_configured; then
  echo "/etc/rc.conf is not configured.  Multiuser boot aborted."
  exit 1
fi

if [ "$1" = autoboot ]; then
  autoboot=yes
  _rc_fast_run=yes          # run_rc_command(): do fast booting
fi

stty status '^T'

#       Set shell to ignore SIGINT (2), but not children;
#       shell catches SIGQUIT (3) and returns to single user.
#
trap : 2
trap "echo 'Boot interrupted.'; exit 1" 3

files=`rcorder -s nostart /etc/rc.d/*`

for i in $files; do
  run_rc_script $i start
done

date
exit 0
```

Listing 14.1: Die normale rc-Datei von NetBSD 1.6

Die möglichen Skripte, die über einen Eintrag in der »/etc/rc.conf« gestartet werden, befinden sind im Verzeichnis »/etc/rc.d«. Jedes Skript in diesem Verzeichnis gehört genau zu einem Dienst beziehungsweise löst eine Aktion aus[1].

```
#       $NetBSD: rc.conf,v 1.85.2.9 2001/04/24 22:42:44 he Exp $
#
# see rc.conf(5) for more information.
#
# Use program=YES to enable program, NO to disable it. program_flags are
# passed to the program on the command line.
#

# Load the defaults in from /etc/defaults/rc.conf (if it's readable).
# These can be overridden below.
#
if [ -r /etc/defaults/rc.conf ]; then
  . /etc/defaults/rc.conf
fi

# If this is not set to YES, the system will drop into single-user mode.
#
rc_configured=YES

# Add local overrides below
#
```

Listing 14.2: Die rc.conf-Datei enthält die benutzerdefinierten Konfigurationsdaten und ist anfangs praktisch leer

Ist *autoboot* auf *YES* gesetzt, wird zusätzlich verhindert, daß die zu startenden Skripte im Verzeichnis »/etc/rc.d« versuchen, zu überprüfen, ob ein Prozeß bereits läuft. Damit wird die Zeit, die für das Hochfahren des Systems benötigt wird, verringert. Allerdings erfolgt diese Prüfung nicht, wenn vorher bereits der Einzelbenutzermodus aktiv war.

Danach wird mit Hilfe des Programms *rcorder* eine Startreihenfolge für die Skripte in »/etc/rc.d« festgelegt. Dabei werden alle Skripte bestimmt, bei denen das Schlüsselwort *nostart* ungültig ist (das sind diejenigen, bei denen das Schlüsselwort im Skript fehlt). Das Resultat wird einer Variable zugewiesen, die anschließend ausgewertet wird. Bei jedem so bestimmten Skript wird als erster Parameter der String *start* übergeben. Das entsprechende Skript wird in einer Sub-Shell ausgeführt, es sei denn, sein Name endet

[1] Es ist eine Frage, wie man einen »Dienst« definiert. Zum Samba gehören beispielsweise drei Aufrufe, die logischerweise auch über drei Skripte geregelt werden. Es handelt sich aber nur zwei Daemons, die mit diesen drei Aufrufen konfiguriert und gestartet werden.

mit ».sh«. In diesem Fall erfolgt der Start in der aktuellen Shell. Beachten Sie, daß das Skript den Wert 0 zurückgeben muß. Ansonsten wird das Hochfahren des Systems beendet (und man landet bestenfalls im Einzelbenutzermodus)!

14.2 /etc/rc.shutdown

Das Skript »/etc/rc.shutdown« ist prinzipiell das Gegenstück zur »/etc/rc«. Es wird immer dann aufgerufen, wenn die einzelnen Dienste heruntergefahren oder neu gestartet werden sollen. In der Regel ruft das Kommando *shutdown* das Skript vor dem Herunterfahren des Systems auf.

Auch in der »/etc/rc.shutdown« werden die beiden Skripte »/etc/rc.subr« und »/etc/rc.conf« eingebunden. Damit stehen ebenfalls die bereits bei der »/etc/rc« angesprochenden Funktionen zur Verfügung. Danach wird ebenfalls das Programm *rcorder* aufgerufen. Jetzt werden die Skripte bestimmt, bei denen das Schlüsselwort *shutdown* gültig ist (es steht direkt im Skript). Zusätzlich wird die Reihenfolge in der Liste umgekehrt, so daß das letzte mit »/etc/rc« gestartete Skript zuerst beendet wird (sofern das Schlüsselwort *shutdown* in der Datei steht).

Das Resultat wird einer Variaben zugewiesen, die anschließend entsprechend ausgewertet wird. Jedem aufzurufendem Skript wird als erster Parameter der String *stop* übergeben und es wird in einer Sub-Shell gestartet. Endet der Name mit ».sh«, erfolgt auch hier die Ausführung in der aktuellen Shell.

14.3 Der Start des xdm

Wie das init-System aufgebaut wird, sieht man recht gut am Display-Manager *xdm*. Um ihn zu starten, muß die Variable *xdm* in der Konfigurationsdatei »/etc/rc.conf« auf den Wert *YES* gesetzt sein. Fährt man nach dem Eintragen des Werts das NetBSD-System neu hoch, sieht man den Display-Manager auf dem Bildschirm und kann sich entsprechend anmelden.

Durch die Angabe von *YES* nach der Variable führt der Init-Prozeß das Skript »/etc/rc.d/xdm« aus (Listing 1). Da die Datei keine Endung ».sh« besitzt, wird dafür eine Sub-Shell gestartet. Die Datei beginnt mit eine Reihe von Kommentarzeilen. Darin werden eine Versionsnummer festgelegt und man kann ablesen, wann das Skript erzeugt wurde. Wichtig sind die nächsten drei Zeilen, in denen die Worte *PROVIDE*, *REQUIRE* und *KEYWORD* zu finden sind (außerdem ist die Großschreibung zu beachten). Mit *PROVIDE* wird festgelegt, um welches Skript es eigentlich geht. *REQUIRE* gibt an, zu welchem Zeitpunkt das Skript gestartet werden soll. Die Angabe von *DAEMON* in der Liste der Optionen definiert, daß es vor allen Daemons aufgerufen werden soll, die allgemeinen Zwecken dienen. Weiterhin wird es vor den Anmelde-Prozessen der Benutzer gestartet (*LOGIN*). Allerdings benötigt es für ein korrektes Arbeiten den Treiber *wscons*, so daß es erst nach ihm aufgerufen werden kann. Diese Informationen braucht übrigens das zuvor angesprochene Programm *rcorder*, um eine Startreihenfolge festzulegen.

```
#!/bin/sh
#
# $NetBSD: xdm,v 1.4.4.2 2000/10/02
# 01:09:52 lukem Exp $
#

# PROVIDE: xdm
# REQUIRE: DAEMON LOGIN wscons
# KEYWORD: shutdown

. /etc/rc.subr

name="xdm"
rcvar=$name
command="/usr/X11R6/bin/${name}"
pidfile="/var/run/${name}.pid"
required_files="/usr/X11R6/lib/X11/xdm\
/xdm-config"
extra_commands="reload"

load_rc_config $name
run_rc_command "$1"
```

Listing 14.3: Die Datei »/etc/rc.d/xdm«

Danach erfolgt die Angabe eines Schlüsselworts (*KEYWORD*), auf das die Datei beispielsweise durch das Shutdown-Skript »rc.shutdown« hin untersucht wird. Die Angabe von *shutdown* in diesem Zusammenhang bedeutet, daß das Skript auch beim Herunterfahren aufgerufen werden sollte.

In der nächsten Zeile werden die Unterfunktionen aus der Datei »/etc/rc.subr« geladen. Wie alle Start- und Stopskripte macht auch *xdm* davon Gebrauch. Damit die Funktionen aber korrekt ausgeführt werden können, müssen zuerst einmal einige Variablen gesetzt werden. Dazu gehören der Name (*name*) und das auszuführende Kommando (*command*), das natürlich mit dem Namen verknüpft ist. Die Zeile *rcvar* dazwischen gibt an, ob die Aktion ausgeführt werden soll. Mit ihr kann zusätzlich kontrolliert werden, ob bei einem Start überprüft wird, ob ein solcher Prozeß bereits läuft. Wenn der Rechner das erste Mal hochgefahren wird, ist diese Überprüfung natürlich unnötig, so daß sie in diesem Fall ausgeschaltet wird.

Zusätzlich wird mit *pidfile* festgelegt, an welcher Stelle im Dateisystem eine Datei mit der Prozeß-ID gefunden werden kann oder wo der Prozeß sie anlegt. Die Datei wird auf ihr Vorhandensein hin überprüft. Benötigt der Prozeß weitere Dateien, um korrekt zu funktionieren, lassen sich diese mit der Variablen *required_files* festlegen. In der Regel werden hierbei meist Konfigurationsdateien angegeben, wie in diesem Beispiel auch. Gibt es diese Dateien nicht oder sind sie nicht lesbar, bricht die Ausführung ab.

Die Variable *extra_commands* legt schließlich noch fest, welche zusätzlichen Befehle außer dem Standardsatz *start* (Aufruf des Skripts zum Starten), *stop* (Aufruf des Skripts zum Beenden), *restart* (Aufruf des Skripts zum Stoppen und zum erneuten Starten) und *status* (Ausgabe des Status, wobei in der Regel nur die Prozeß-ID angezeigt wird) dem Skript noch übergeben werden können. Die Angabe von *reload* bedeutet in diesem Fall, daß der *xdm* statt eines *SIGTERM*- ein *SIGHUP*-Signal erhält. Dadurch werden nur die Konfigurationsdateien neu gelesen und auf Veränderungen hin überprüft. Dieses Verhalten gilt hier speziell nur für den *xdm*, bei anderen Daemonen, die das Kommando *reload* unterstützen, kann es durchaus anders sein.

Damit sind die Voraussetzungen geschaffen, um die eigentliche Aktion ausführen zu können, den Display-Manager *xdm* als Prozeß zu starten. Davor wird aber noch eine eventuell vorhandene Konfigurationsdatei in die aktuelle Umgebung geladen (*load_rc_config*). Nach dieser Datei wird meist im Verzeichnis »/etc/rc.conf.d« gesucht, aber wenn Sie sich das Verzeichnis genauer anschauen, werden Sie feststellen, daß es leer ist. Die Zeile befindet sich in den meisten Skripten, bewirkt aber nicht, daß die Abarbeitung des Skripts beendet wird (im Gegensatz zu *required_files*). Erst danach folgt mit dem Befehl *run_rc_command* der eigentliche Aufruf des Display-Managers *xdm* (in diesem Beispiel). Beachten Sie, daß dem Kommando nicht(!) der Name des Display-Managers übergeben wird, sondern das Argument, das als erstes beim Aufruf des Skripts »/etc/rc.d« angegeben wurde. Beim Hochfahren des *xdm* dürfte das der String *start* sein, so daß damit dann das Kommando ausgeführt wird, das bei der Variable *command* gesetzt ist.

Insgesamt sieht das Ganze komplizierter aus, als es wirklich ist. Sie müssen nur dreimal um die Ecke denken und schon wissen Sie, wie es funktioniert. Ein wenig mehr Direktheit würde das Verständnis des Startens beispielsweise der Daemonen fördern, aber das NetBSD-System scheint ja auch mit diesem Konzept gut zu funktionieren.

Funktion	Beschreibung
checkyesno VARIABLE	Der Parameter wird daraufhin getestet, ob *VARIABLE* gleich *YES* oder *NO* ist. Handelt es sich um einen anderen String, wird eine Warnung ausgegeben. Die Funktion gibt bei einem *yes* den Wert 0 zurück, ansonsten einen Wert ungleich 0.
check_pidfile PIDDATEI PROCNAME	Überprüft die erste Zeile einer *PIDDATEI* und vergleicht die PID mit der des Prozesses *PROCNAME*. Ist das erfolgreich, wird die zugehörige PID ausgegeben.
check_process PROCNAME	Überprüft, ob einer oder mehrere Prozesse mit dem Namen *PROCNAME* laufen. Ist das der Fall, wird eine Liste mit den zugehörigen PIDs ausgegeben.

Tabelle 14.1 (Fortsetzung auf nächster Seite)

Funktion	Beschreibung
err WERT MELDUNG	Die angegebene *MELDUNG* wird auf *stderr* ausgegeben und im *syslog* protokolliert. Anschließend wird das Skript mit *WERT* beendet.
load_rc_config DATEI	Lädt die zu *DATEI* passende Konfigurationsdatei in die aktuelle Umgebung (meist aus dem Verzeichnis »/etc/rc.conf.d/«).
mount_critical_filesystems	Die Funktion untersucht die Liste mit den kritischen Dateisystemen und überprüft, welche schon eingebunden sind. Alle anderen werden eingebunden.
rc_usage KOMMANDO	Gibt eine Liste mit gültigen Kommandos aus.
run_rc_command ARGUMENT	Nach dem String *ARGUMENT* wird in der Liste der unterstützten Kommandos (*rcvar, reload, restart, start, status, stop,* ${*extra_commands*}) gesucht. Wird ein e Übereinstimmung gefunden, wird entweder das Kommando ${*ARGUMENT*}*_cmd* oder das Standardkommando ausgeführt. Dieser Aufruf wird in verschiedenen rc-Skripten verwendet. Dabei werden einige globale Variablen benutzt, die vor der Verwendung von *run_rc_command* in der entsprechenden Datei gesetzt werden können. Besitzt *ARGUMENT* den Präfix *fast*, wird die Überprüfung der PID übersprungen, bei einem *force* wird die Variable ${*rcvar*} auf *YES* gesetzt. Entspricht ARGUMENT dem Kommando *status*, wird angezeigt, ob ${*command*} läuft. Bei einem *start* wird überprüft, ob ${*command*} bereits läuft und ob ${*rcvar*} auf *yes* gesetzt ist. In diesem Fall wird ${*command*} ausgeführt. Das Standardkommando für *ARGUMENT* = *stop* überprüft, ob die Variable ${*pidfile*} gesetzt ist. In diesem Fall wird dem Prozeß mit der zugehörigen PID ein SIGTERM-Signal gesendet. Im anderen Fall erhält der Prozeß ${*command*} das SIGTERM-Signal. Wird statt *stop* das Kommando *reload* verwendet, erhalten die Prozesse kein SIGTERM-, sondern ein SIGHUP-Signal. Eine weitere Möglichkeit für *ARGUMENT* ist *restart*. In diesem Fall werden die Aktionen nacheinander ausgeführt, die beim *stop*- und beim *start*-Kommando beschrieben sind.

Tabelle 14.1 (Fortsetzung auf nächster Seite)

Funktion	Beschreibung
run_rc_script DATEI ARGUMENTE	Startet das Skript mit dem Namen *DATEI* und den Parametern *ARGUMENTE*. Außerdem wird sichergestellt, daß die Rückgabewerte korrekt weiterverarbeitet werden. Endet der Name der Datei auf ».sh«, wird die aktuelle Umgebung zur Ausführung verwendet. Im anderen Fall erfolgt die Ausführung als Unterprozeß. Wird ein ».sh«-Skript verwendet, sollte das Kommando *run_rc_command* nicht verwendet werden, da nicht sichergestellt ist, daß das Skript den Status der globalen Variablen nicht ändert. Eine Änderung würde auch andere Skripte betreffen, so daß die Resultate nicht vorhersagbar sein können.
warn MELDUNG	Die angegebene *MELDUNG* wird auf *stderr* ausgegeben und im *syslog* protokolliert.

Tabelle 14.1: In der Datei »/etc/rc.subr« werden einige Shell-Funktionen definiert, die beispielsweise in der Datei »/etc/rc« und den Startskripten zum Einsatz kommen

Globale Variable	Beschreibung
command	Definiert den vollen Namen des auszuführenden Befehls. Der Befehl selbst muß nicht angegeben werden, wenn eine Datei mit dem Namen *${ARGUMENT}_cmd* für jedes Schlüsselwort vorhanden ist.
command_args	Legt optionale Argumente oder Shell-Direktiven für den Befehl fest.
extra_commands	Eine Liste mit zusätzlichen Kommandos, die unterstützt werden.
name	Der Name des Skripts. Diese Variable muß gesetzt werden.
pidfile	Ist diese Variable gesetzt, wird die Funktion *check_pidfile* ausgeführt. Ansonsten wird meist *check_process* verwendet.
rcvar	Diese Variable legt fest, ob die Aktion ausgeführt werden soll (bei einem *yes*).
required_dirs	Die Existenz der angegebenen Verzeichnisse wird überprüft, bevor das Standardkommando ausgeführt wird.
required_files	Die Lesbarkeit der spezifizierten Dateien wird vor der Ausführung des Standardkommandos unter die Lupe genommen.

Tabelle 14.2 (Fortsetzung auf nächster Seite)

Globale Variable	Beschreibung
required_vars	Die Funktion *checkyesno* wird für jede Variable in der angegebenen Liste ausgeführt. Sind alle vorhanden, wird das Standardkommando ausgeführt.
${name}_chdir	Gibt ein Verzeichnis an, in das vor der Ausführung von *${command}* gewechselt wird.
${name}_chroot	Legt ein neues *root*-Verzeichnis für die Ausführung von *${command}* fest.
${name}_flags	Gibt Argumente an, die *${command}* übergeben werden. Diese können aber durch die allgemeine Umgebungsvariable *$flags* überschrieben werden.
${name}_group	Gruppe, unter der das Kommando *${command}* ausgeführt wird, sofern ein *${name}_chroot* angegeben ist.
${name}_groups	Liste mit zusätzlichen Gruppen, unter denen das Kommando *${command}* ausgeführt wird, wenn *${name}_chroot* angegeben ist.
${name}_nice	Stufe, in der *${command}* ausgeführt wird.
${name}_user	Bezeichnet den Benutzer, unter dem *${command}* ausgeführt wird. Dabei wird das Kommando *su* verwendet, sofern nicht *${name}_chroot* angegeben ist.
${_arg}_cmd	Ist diese Variable gesetzt, wird die zugehörige Aktion ausgeführt.
${_arg}_precmd	Bevor die Hauptaktion gestartet wird, wird das hiermit gesetzte Programm ausgeführt. Gibt es einen Fehler zurück (Wert ungleich 0), wird das Programm *${_args}_cmd* nicht ausgeführt.

Tabelle 14.2: Die von run_rc_command genutzten globalen Variablen

15 Administration mit Webmin

von Georg Basse

Die Administration von Unix-Systemen und ihrer Applikationen wie Apache oder Sendmail gilt vielerorts als Buch mit sieben Siegeln. Dabei sind die Siegel leichter zu brechen als oftmals angenommen – vorausgesetzt, der frischgebackene Systemadministrator nutzt die Hilfestellung von Webmin, eines universellen und zudem als Open-Source-Programm vertriebenen Werkzeugs.

Webmin ist ein Tool für Unix-Administratoren zur webbasierenden Konfiguration von Diensten, wie beispielsweise für Webserver (Apache), Mailserver (Sendmail) oder Proxy (Squid). Auch die Administration von Eigenschaften des Betriebssystems ist möglich, zum Beispiel das Einrichten und Bearbeiten von Unix-Benutzern oder die Konfiguration der Netzwerk-Interfaces. Das hauptsächlich auf Perl basierende Tool ist einfach zu installieren und zu konfigurieren. Durch seinen modularen Aufbau wird das Einbinden neuer oder selbst entwickelter Funktionen erleichtert. Die Steuerung per Browser ermöglicht den Zugriff von jedem Arbeitsplatz aus, ohne daß eine lokale Installation irgendwelcher Programme notwendig wäre.

Webmin ist derzeit für viele unterschiedliche Unix-Derivate verfügbar. Damit ist die Konfiguration sowohl einzelner Server als auch einer gesamten Serverlandschaft mit verschiedenen Unix-Plattformen unter einer einheitlichen Oberfläche möglich. Je nach verwendetem Betriebssystem unterscheidet sich die Anzahl und Art der verfügbaren Webmin-Standardmodule jedoch deutlich.

Die Konfiguration von Diensten und Betriebssystemeigenschaften erfolgt ausschließlich in den jeweiligen Konfigurationsdateien dieser Softwarekomponenten. Diese Tatsache macht die über Webmin vorgenommenen Anpassungen nachvollziehbar, bereits vorhandene Einstellungen sind in den Webmin-Masken sichtbar. Eine eigene User- und Rechteverwaltung erlaubt es zudem, für verschiedene User unterschiedliche Module freizugeben.

15.1 Historie

Das Webmin-Projekt wurde von Jamie Cameron, der übrigens trotz Namensähnlichkeit zu Jamie Lee Curtis männlichen Geschlechts ist, mit der ersten Veröffentlichung (Version 0.1) im Oktober 1997 begonnen. In den vergangenen Jahren wurden in bester Open-Source-Manier zahlreiche Versionen über den zentralen Server (*http://www.webmin.com*) und seine Mirrors verbreitet. Die zum Zeitpunkt der Drucklegung dieses Buchs aktuelle Version 1.030 stammt vom November 2002. Während Sie diesen Text lesen, ist diese Version jedoch gewiß bereits durch ein neueres Release abgelöst, denn die Entwicklung und Verbesserung geschieht mit großer Geschwindigkeit.

Die Entwicklung von Webmin verlief bezüglich seiner Zielsetzung und Lizenzpolitik durchaus stürmisch. Ursprünglich hatte der Autor das Programm unter eine GPL-ähnliche Open-Source-Lizenz gestellt und angekündigt, die Entwicklung ab der Version 1.0 in eine kommerzielle und eine freie Linie aufzuspalten. Im Winter 1999 verkaufte er dann überraschend die Rechte an seiner Entwicklung an den amerikanischen Linux-

Distributor Caldera Inc. Dies wurde auf der Mailingliste des Projekts von heftigen Wortgefechten begleitet, da sich viele Anwender und Mitentwickler getäuscht sahen und eine ungewisse Zukunft der freien Entwicklung befürchteten. Caldera hat Webmin jedoch unter eine BSD-artige Lizenz gestellt, die sowohl kommerzielle als auch nicht kommerzielle Nutzung, Weiterentwicklung und Vertrieb gestattet. Nun ist Webmin sogar freier als bisher. Das war übrigens auch nicht anders möglich, da es auf Perl basiert und dessen Lizenzmodell unterworfen ist. Hier zeigt sich, wie Open-Source-Lizenzen die kontinuierliche Produktentwicklung sichern können. Cameron ist mittlerweile für Caldera tätig und nach wie vor sehr engagiert an der Weiterentwicklung von Webmin beteiligt.

Für Caldera spielte das Programm lange eine zentrale Rolle, besonders in der als eServer vertriebenen Variante seiner Linux-Distribution. Dort diente es als das zentrale Tool zur Administration wichtiger Dienste und Systemkomponenten und trat in Wettbewerb mit den Tools anderer Distributionen, zum Beispiel LinuxConf von RedHat-Linux oder YaST der SuSE AG. Webmin hat sich dort bewährt und mittlerweile auch Einzug in das Gemeinschaftsprojekt »United Linux« von SuSE, Connectiva, Caldera und Turbo Linux gehalten. Doch der Anspruch von Webmin, auch komplexe Applikationen bequem konfigurierbar zu machen, beschränkt sich nicht nur auf Linux. Vielmehr wurde es auf eine ansehnliche Reihe freier und kommerzieller Unix-Plattformen portiert, auch auf NetBSD.

15.2 Aufbau und Arbeitsweise

Möglich ist die weite Verfügbarkeit von Webmin nicht zuletzt dadurch, daß es ausschließlich in Perl 5 ohne die Verwendung externer Perl-, C/C++- oder anderer Module realisiert wurde. Prinzipiell läuft es daher auf allen Plattformen, auf denen eine Standard-Perl5 Distribution erhältlich ist – eine Voraussetzung, die NetBSD natürlich erfüllt.

Die Perl-Module arbeiten im wesentlichen als CGI-Programme eines Webservers, wodurch das Benutzerinterface in HTML zu realisieren war und Webmin geradezu zwangsläufig voll netzwerktauglich ist. Der mitgelieferte Webserver des Programms ist ebenfalls in Perl 5 geschrieben und umfaßt weniger als tausend Programmzeilen. Sofern das SSLeay-Modul für Perl installiert wurde, kann er auch komplett über Secure-HTTP (HTTP-S) statt HTTP arbeiten, Browser und Server kommunizieren also abhörsicher über eine verschlüsselte Verbindung.

Wer diesem internen Webserver nicht traut, kann beispielsweise auch mit Apache einen virtuellen Server aufsetzen und Webmin auf diese Weise benutzen. Allerdings büßt man dabei die interne Benutzerverwaltung und die umfangreichen Konfigurationsmöglichkeiten der Oberfläche selbst ein – doch dazu später mehr.

Wie eingangs erwähnt, erfolgen alle durch Webmin vorgenommenen Änderungen direkt in den jeweiligen Konfigurationsdateien der Applikationen und Systemkomponenten, es gibt keine Zwischenformate, Datenbanken oder ähnliches. Dadurch bleiben webmin-administrierte Systeme offen für manuelle Änderungen, bereits mühevoll per Editor geschaffene Einstellungen bleiben erhalten. Das ist besonders dann ein Vorteil, wenn man anhand der webmin-eigenen Hilfe zu einem Modul nicht mehr weiter weiß.

So kann man sich anhand einer Manpage oder anderer Dokumentationen in die Konfiguration eines Programmes einarbeiten. Die manuellen Änderungen finden sich dann sofort auch in den Webmin-Masken wieder.

Wie auf *http://www.webmin.com* zu sehen ist, kann Webmin auf einer Reihe freier und kommerzieller Unix-Systeme eingesetzt werden. Unterschiede bestehen auf den verschiedenen Plattformen in der Art und Anzahl der Module. Zwar können alle Perl-Module auf jeder Plattform ausgeführt werden, was jedoch nicht bedeutet, daß diese überall auch sinnvolle Dienste leisten können – eine LILO-Konfiguration auf einer Hewlett-Packard-PA-RISC-Maschine unter HP-UX ist natürlich zum Scheitern verurteilt.Seit der Version 0.76 gliedert Webmin die Module in einzelne Gruppen auf, die in Form von »Karteikarten« auf der Startseite dargestellt werden. Seit Version 0.8x ist Webmin durch Stilvorlagen (Themes) im Aussehen anpaßbar. Die Darstellung der Gruppen ist vom jeweiligen Theme abhängig, hier wurde das Default-Theme von MSC Linux benutzt.

Ein Grund für den großen Leistungsumfang von Webmin ist das Vorhandensein einer gut dokumentierten Programmierschnittstelle (API), die es erlaubt, dem System eigene Module hinzuzufügen. Eine Beschreibung der Schnittstelle ist in verschiedenen Textformaten, darunter auch als PDF-Datei für den Acrobat-Reader vom Webmin-Server zu beziehen. Dort ist auch ein vorbereitetes Template-Archiv zu finden, das alle notwendigen Verzeichnisse und Dateien enthält, die ein Modul für Webmin benötigt. Mit durchschnittlichen Perl-Kenntnissen ausgestattet, kann jeder Programmierer so neue Module erstellen. Webmin erleichtert dies mit einer ganzen Reihe von Funktionen für die Ausgabe von Standard-HTML-Elementen, aber auch für die Prozeßverwaltung, die Implementation eines Berechtigungskonzepts mit ACLs, das Arbeiten mit Datenfeldern und den Zugriff auf Dateien. Die API-Dokumentation gibt auch Tips für ein einheitliches Look&Feel und beschreibt Besonderheiten bei der Erstellung plattformabhängiger Module. Hierbei sind auch die vorhandenen Module einen genauen Blick wert, da sie den Einsatz der Funktionen am besten demonstrieren.

15.3 Installation

Webmin ist nicht in der NetBSD-Grunddistribution enthalten und auch nicht in der NetBSD-Softwareliste des NetBSD-Projekts (*http://www.netbsd.org/*) aufgelistet. Es läßt sich jedoch von der Homepage (*http://www.webmin.com*) herunterladen. Dort sind verschiedene Paketformate verfügbar; für NetBSD ist das gepackte Tar-Archiv Paketformat der Wahl, ein Package ist nicht verfügbar.

Zusätzlich muß Perl 5 auf dem System installiert sein, das für NetBSD 1.6 verfügbare Package der Version 5.6.1 wurde für alle diesem Kapitel zugrunde liegenden Schritte verwendet. Wollen Sie, wie an dieser Stelle ausdrücklich aus Sicherheitsgründen empfohlen, eine verschlüsselte Kommunikation zwischen Server und Browser aufbauen, ist zusätzlich auch das passende SSLeay Paket vonnöten.

Für die Installation sollte man als Systemverwalter (*root*) angemeldet sein. Ist Perl 5 installiert und die Datei »webmin.tar.gz« vorhanden, kann mit der Webmin-Installation begonnen werden.

Abb. 15.1: Die Modulgruppen von Webmin, hier im grafisch aufgepeppten Stil

1. Das Archiv »webmin-1.030.tar.gz« befindet sich in diesem Beispiel unter »/usr/local«.
2. Mit *tar -xvzf webmin-1.030.tar.gz* wird das Archiv in das aktuelle Verzeichnis entpackt.
3. Nun wechseln wir in das Verzeichnis »/usr/local/webmin-1.030« und führen die Datei »setup.sh« aus (*./setup.sh*).
4. Die erste Meldung lautet *Config file directory {/etc/webmin}:* Hier geben wir das Verzeichnis an, in das die Konfigurationsdateien von Webmin gelegt werden sollen; Vorgabe ist »/etc/webmin«.
5. *Log file directory {/var/webmin}:* Fragt nach dem Verzeichnis für Protokolldateien.
6. Jetzt wird der absolute Pfad zu Perl abgefragt. Unter NetBSD ist dies »/usr/pkg/bin/perl«.
7. Webmin arbeitet wie oben beschrieben als Web-Applikation und bringt einen eigenen HTTP-Server mit. Um eine Störung eines eventuell schon installierten HTTP-Servers zu vermeiden, horcht der webmin-eigene Server auf einem anderen TCP-Port als ein normaler Webserver. Die Vorgabe (Port 10000) kann normalerweise übernommen werden (*Web server port {default 10000}*).
8. Zur Administration wird ein Webmin-Benutzer mit allen Rechten angelegt. Hier können Sie nun den Namen und anschließend das Paßwort vergeben. Vorschlag für den Namen ist *admin*, dieser kann aber beliebig geändert werden.
9. Nun ist das Paßwort einzugeben und durch eine zweite Eingabe zu bestätigen.
10. Als letztes ist noch anzugeben, ob der Webmin-Server bei jedem Neustart mit gestartet werden soll. Das Skript trägt bei einer Bejahung dieser Frage den Startaufruf für Webmin in die »/etc/rc.conf« ein.

Nach der erfolgreichen Installation kann mit dem Webbrowser des Vertrauens auf Port 10000/tcp des NetBSD-Rechners zugegriffen werden (also *http://cynar.liquernet.de:10000* für einen NetBSD-Rechner namens Cynar). Nach der Anmeldung mit den oben definierten Daten begrüßt Webmin mit einer Gruppenleiste wie in Abbildung 15.1 und zeigt die Module der ersten Gruppe an.

Im folgenden soll Webmin nicht vordergründig auf Basis dieser Gruppen betrachtet werden, sondern anhand der Bedeutung der einzelnen Module im administrativen Tagesgeschäft.

15.4 Starten und Stoppen von Webmin

Die Webmin-Konfiguration erlaubt es, wesentliche Einstellungen an Webmin selbst vorzunehmen. Dazu gehören folgende Funktionen:

1. *IP Zugangskontrolle:* Diese Funktion stellt ähnlich dem TCP-Wrapper die Möglichkeit bereit, den Zugriff auf Webmin nur von bestimmten Netzknoten aus zu erlauben. Dies ist ein wichtiges Sicherheitsfeature bei großen Netzen. Mögliche Regeln sind:
   - *Gestatte Zugriff von allen Adressen* ist die Standardeinstellung. So kann von jeder Arbeitsstation aus auf Webmin zugegriffen werden.
   - *Gestatte Zugriff nur von aufgelisteten Adressen*: Nur die aufgelisteten IP-Adressen können eine Verbindung mit Webmin aufbauen.
   - *Verweigern von aufgelisteten Adressen*: Hier werden nur die in die Liste eingetragenen Adressen abgewiesen.
2. *Port und Adresse:* Hier kann der bei der Erstinstallation angegebenen TCP-Port und die Adresse des Webminservers nachträglich verändert werden.
3. *Logging:* Webmin läßt sich so konfigurieren, daß er alle Zugriffe in einer Logdatei protokolliert (Standard-CRLF-Logdatei-Format). Sie können wählen, ob IP-Adressen oder Hostnamen protokolliert werden sollen. Ebenfalls können Sie angeben, wie oft die Logdatei bereinigt (gelöscht) werden soll. Die Logdatei wird nach »/var/webmin/miniserv.log« geschrieben und kann mit gängigen Tools wie dem Webalizer ausgewertet werden.
4. *Proxy Server:* Sollte eine Firewall den Webmin-Server vom Internet trennen, kann dieser das Internet eventuell nur über einen Proxy-Server erreichen. Die Module, die einen Zugriff über HTTP auf das Internet vornehmen, werden dann über diesen Proxy darauf zugreifen (zum Beispiel ein Download/Upgrade von Modulen oder einer neuen Webmin-Version).
5. Farbschema: Mit diesem Modul können Sie das Farbschema des gesamten Webmin verändern. Jede Farbe wird als RGB-Wert beschrieben, sie ist als Hexadezimalzahl (00 bis FF) anzugeben.
6. *Webmin-Module:*
   - *Installieren:* Webmin-Module können jederzeit hinzugefügt oder gelöscht werden. Die Module haben normalerweise die Dateiendung »*.wbm«. Jede dieser Dateien kann ein oder mehr Module enthalten.
   - *Duplizieren/Klonen:* Mit dieser Funktion ist das Duplizieren oder Klonen von Modulen möglich. Sollten Sie mehr als eine Kopie eines Moduls mit unterschiedlichen Einstellungen benutzen wollen, ist das Klonen eines Moduls die Lösung. Ein Klon erhält alle Eigenschaften eines Originals.
   - *Löschen:* Die Module und ihre Kopien können auch wieder gelöscht werden. Sollten Sie ein Modul löschen, das einen Klon besitzt, wird dieses ebenfalls gelöscht. Das Löschen eines Moduls mit einer Abhängigkeit zu einem anderen ist nicht möglich.

7. *Betriebssystem:* Unter diesem Punkt kann die Definition des Betriebssystems geändert werden, das der Webmin steuern soll. Für Betriebssystem-Updates ist dies eine nützliche Funktion. Aber Achtung: Die Einstellungen bereits installierter Module werden nicht angepaßt
8. *Sprache:* Mit dieser Einstellung steuern Sie die Standardsprache für Webmin. Über die Benutzerverwaltung kann das für jeden Anwender auch separat gesteuert werden.
9. *Einstellungen der Startseite:* Die Startseite kann hier auf Ihre bevorzugten Einstellungen, soweit möglich, angepaßt werden.
10. *Webmin-Upgrade:* Dieses Modul gibt Ihnen die Möglichkeit, die Webmin-Installation zu aktualisieren, sowohl aus einer lokalen Datei oder per FTP direkt von der Webmin-Homepage (*www.webmin.com*). Die vorgenommenen persönlichen Einstellungen werden natürlich beibehalten.

15.5 Serverindex

Der Serverindex ist eine Möglichkeit, mehrere Webmin-Server von einer Startseite aus erreichbar zu machen, was zur Administration von Serverfarmen sinnvoll sein kann. So ist eine Zentralisierung des Zugriffs auf andere Server leicht realisierbar. Der Index wird auf einem zentralen Server angelegt, nur dieser Server muß die weiteren Server erreichen können. Firewall-Administratoren wird dies freuen, da eine einzige Firewallregel für den Zugriff auf die gesamte Serverlandschaft genügt.

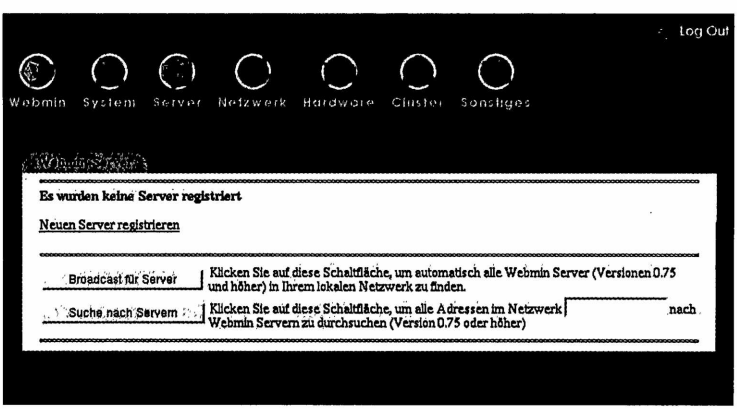


Abb. 15.2: Der Serverindex zentralisiert den Zugriff auf eine webmin-administrierbare Serverfarm

So wird ein Eintrag angelegt:

1. *Neuen Server registrieren* anwählen.
2. In das Feld *Hostname* den Hostnamen des zu erreichenden Servers eintragen. Sollte keine DNS-Auflösung verfügbar sein, kann auch die IP-Adresse eingetragen werden.
3. Unter *Port* muß die Portnummer eingetragen werden, auf der der Webmin-Server der zu erreichenden Maschine läuft (Vorgabe bei der Installation war hier 10000).
4. Der *Server-Typ* ist das Betriebssystem, mit dem der zu erreichende Server betrieben wird.
5. *Secure Socket Layer Server*: Ist der zu erreichende Server für SSL konfiguriert, ist eine verschlüsselte Kommunikation mit Webmin möglich.
6. *Link Typ*: Bei Auswahl *Normaler Link* erscheint für diesen Server eine Paßwortabfrage beim Verbindungsaufbau. Andernfalls können der Benutzername und das Paßwort hinterlegt werden, es wird dann direkt eine Verbindung mit diesen Angaben aufgebaut (ohne Paßwortabfrage).
7. Mit der Funktion *Finde Server* wird eine Suche veranlaßt, die im erreichbaren Netzwerk nach aktiven Webminservern sucht.

Beim Hantieren mit Webmin muß stets bedacht werden, daß alle Aktionen auf dem System mit root-Rechten durchgeführt werden. Dies ist nötig, da direkt in den Konfigurationsdateien der einzelnen Programme und Systemkomponenten gearbeitet wird. Es ist daher nicht anzuraten, Webmin ohne weitere Vorkehrungen unerfahrenen Benutzern an die Hand zu geben.

Aus diesem Grund kennt Webmin ein eigenes Benutzerkonzept, das es erlaubt, innerhalb des Tools gültige User zu definieren. Für jeden Benutzer lassen sich einzelne Module freigeben oder sperren. Ist ein Benutzer erst einmal angelegt, können für jedes freigegebene Modul getrennt Zugriffsrechte auf Teilfunktionen bestimmt werden. Die Liste der Module für einen Benutzer ist daher mit Links auf die jeweiligen Eingabemasken hinterlegt. So läßt sich recht genau bestimmen, was ein Webmin-Benutzer im System tun darf und was nicht. Wir werden die möglichen Restriktionen für die Module mit dem jeweiligen Modul beschreiben.

So kann ein Systemverwalter Teilaufgaben des administrativen Tagesgeschäfts auch an weniger sachkundige Helfer (*Operator*) delegieren, ohne um sein System fürchten zu müssen. Bei der Verwaltung einer umfangreichen Serverfarm kann das eine große Hilfe sein. Vergleichbare Möglichkeiten bietet übrigens auch ein kommerziell vertriebenes Tool zur Unix-Administration, nämlich die COSMOS-Suite des Herstellers OSM. Hier finden sich noch eine Reihe von Möglichkeiten hinsichtlich der Definition administrativer Workflows. Ein Administrator kann Aufgaben formulieren, einen Fälligkeitstermin festlegen und diese Operatoren zuordnen. Nach Erledigung der Aufgaben erhält der Administrator eine entsprechende Rückmeldung. Diese Art von Arbeitsorganisation kann Webmin noch nicht unterstützen, aber vielleicht findet sich ja bald ein entsprechendes Modul.

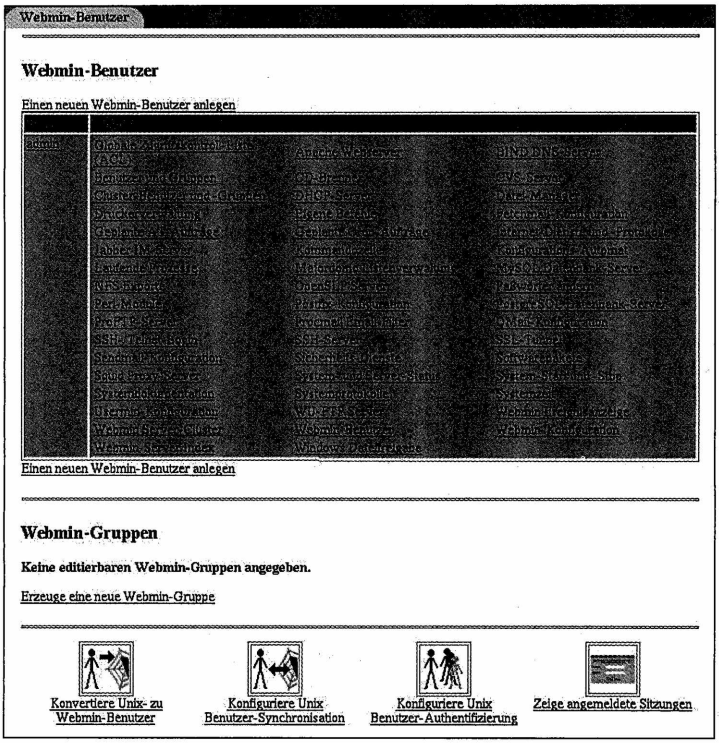


Abb. 15.3: Die webmin-eigenen Benutzer lassen sich einfach anlegen

Bei der Anlage eines neuen Benutzers kann entweder ein bereits definierter Unix-Benutzername verwendet oder ein neuer, nur innerhalb von Webmin gültiger User angelegt werden. Sofern der Benutzer schon unter Unix bekannt ist, läßt sich sein vorhandenes Paßwort übernehmen. Neben der gewünschten Sprache sind dann die jeweiligen Module für diesen User einzustellen. Ist der User einmal angelegt, lassen sich aus der Übersicht heraus die einzelnen Module konfigurieren; dies gilt natürlich auch für die Benutzerverwaltung selbst (siehe Abb. 15.4).

Seit der Version 0.76 läßt sich die Benutzeridentität übrigens auch wechseln, ohne den Browser dazu verlassen zu müssen. Auf der Webmin-Startseite gibt es dazu einen entsprechenden Link.

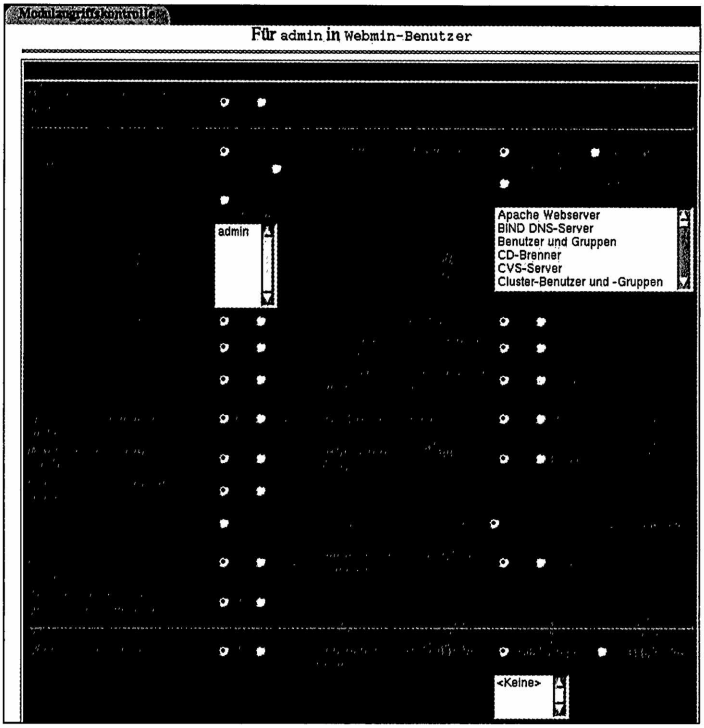


Abb. 15.4: Jedes Webmin-Modul kennt verschiedene Berechtigungsstufen

15.6 Java-Applets

Neben den serverseitig ablaufenden Perl-Modulen bringt Webmin auch zwei Java-Applets mit, die beide auf dem Client ablaufen. Sie finden sich in der Gruppe *Sonstiges*. Einen Browser mit dem entsprechenden Java-Plugin vorausgesetzt, kommt man in den Genuß eines Dateimanagers für den Zugriff auf den Server und eines brauchbaren Telnet-Clients.

Über den Nutzen des Telnet-Clients kann man geteilter Meinung sein. Auf einem Unix-System gehört ein entsprechender Client zum guten Ton und das Java-Applet stellt eher einen Umweg dar. Nutzt man Webmin hingegen von Windows aus, läßt sich eher ein praktischer Nutzen ausmachen.

Im Gegensatz zur microsoft-eigenen Implementation stellt das Applet auch ANSI-Farben korrekt dar und ist im Gegensatz zu Telnet-Clients von Fremdherstellern sofort verfügbar.

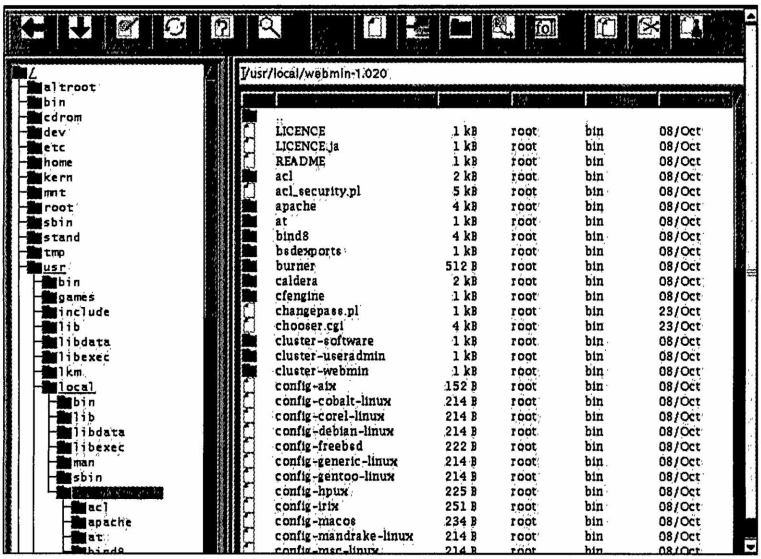


Abb. 15.5: Mit diesem Java-Applet sind Dateien auf dem entfernten Server leicht zu bearbeiten

Der java-basierte Dateimanager hingegen ist ein echtes Highlight, da er alle wesentlichen Dateioperationen auf dem entfernten Server deutlich erleichtert. Dazu gehören Standardoperationen wie das Kopieren, Verschieben, Löschen und Umbenennen von Dateien, aber auch die Änderung der Zugriffsrechte. Ein Doppelklick auf einen Dateinamen zeigt den Dateiinhalt mit Hilfe des Browsers oder eines externen Programms an. Dabei werden die in der Browserkonfiguration definierten Applikationen für die jeweiligen Mime-Typen verwendet. Mit einem eingebauten Minimaleditor können außerdem ASCII-Dateien bearbeitet werden. Wenn also ein direkter Eingriff an einer Konfigurationsdatei auf dem Server nötig sein sollte, kann dies direkt aus Webmin heraus erfolgen.

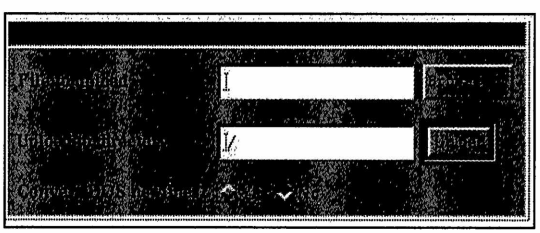


Abb. 15.6: Durch HTTP-Upload gelangen Dateien vom Client auf den Server

Ein ebenfalls sehr nützliches Feature ist der Datei-Upload mit der HTTP-PUT-Methode. Dateien gelangen so auf einfache Weise vom Client auf den Server, ohne hierfür FTP oder NFS bemühen zu müssen.

Natürlich ist dieses Tool nicht unbedenklich, da alle Dateioperationen per Voreinstellung mit Root-Rechten durchgeführt werden. Daran hat auch der Autor gedacht und diesem Modul umfangreiche Einstellungsmöglichkeiten für die Zugriffsrechte spendiert. Für alle wenig privilegierten Webmin-Benutzer ist hier eine Einstellung der Benutzer-ID für Dateioperationen auf den Benutzer *nobody* zu empfehlen. Es können auch nur Teile des Unix-Verzeichnisbaums für den Dateimanager freigegeben werden. Beispielsweise ließe sich für einen Mail-Administrator der Dateizugriff auf die Mailspool-Verzeichnisse einschränken.

15.7 Überwachung

Als ausgesprochen nützlich beim Betrieb beispielsweise von Webservern unter NetBSD hat sich das Modul *System- und Serverstatus* erwiesen, das sich ebenfalls in der Gruppe *Sonstiges* findet. Es prüft bei Aufruf oder auch in regelmäßigen, einstellbaren Abständen die Erreichbarkeit von Serverprozessen wie Apache oder Sendmail und zeigt die Ergebnisse in einer Tabelle an.

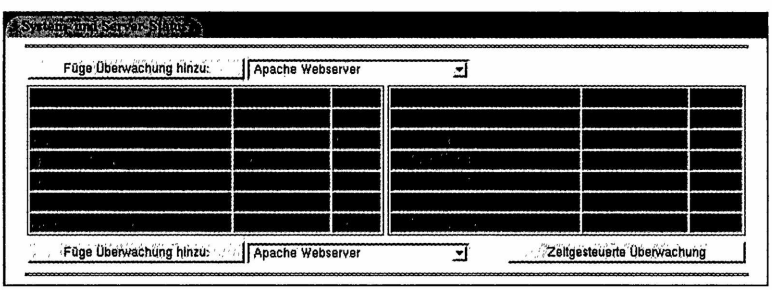


Abb. 15.7: Der System- und Serverstatus hilft bei der Überwachung von NetBSD-Systemen

Fällt ein Serverprozeß aus, wird in der Tabelle eine Markierung gesetzt. Außerdem läßt sich ein externes Programm aufrufen, um eine SMS-Meldung an den Systemverwalter zu senden. Bei kontinuierlicher Überwachung erzeugt Webmin außerdem einen regelmäßigen Statusbericht, der per E-Mail oder SMS verschickt wird. Das Modul ist natürlich keine Konkurrenz für ein echtes System-Management-Framework, dafür aber leicht zu konfigurieren und überall dort bereit, wo Webmin installiert ist.

15.8 Die Benutzerverwaltung

Mit dem Modul *Benutzer und Gruppen* aus der Gruppe *System* von Webmin steht ein komfortables Werkzeug für das Verwalten von Unix-Logins und -Gruppen zur Verfügung. Die vorhandenen Einträge der lokalen Benutzer- beziehungsweise Gruppenda-

tenbank (*»/etc/passwd«* und »/etc/group«) werden tabellarisch angezeigt, Änderungen sind über entsprechende Masken möglich. Nicht verwalten lassen sich per NIS authentifizierte Benutzer, es steht derzeit auch kein NIS-Servermodul hierfür bereit. Praktisch ist beim Anlegen oder Editieren von Benutzern die Möglichkeit, Heimatverzeichnisse automatisch anzulegen oder zu verschieben sowie bei einer Änderung der User-ID diese für alle Dateien im Heimatverzeichnis zu ändern. Paßwörter sind sowohl als Klartext-Paßwörter als auch als bereits verschlüsselte Strings erlaubt, können also zum Beispiel aus einer vorhandenen »/etc/passwd« auf ein anderes System übernommen werden.

Außerdem besteht die Möglichkeit, sich eine Liste der letzten Logins des jeweiligen Users ausgeben zu lassen. So lassen sich »Karteileichen« erkennen, Benutzer also, die zwar noch auf dem System eingetragen sind, aber es nicht mehr benutzen. Auch die Aktivität reiner FTP-Accounts ist so leicht zu überwachen.

Das Modul kennt einen Batchmodus (*Create multiple users*), mit dem sich ausgehend von einer Textdatei mehrere Benutzer anlegen lassen. Dies erspart bei der Übernahme gleicher Benutzer auf verschiedene Systeme das manuelle Editieren der entsprechenden Dateien in »/etc«.

Für jeden Webmin-Benutzer erlaubt die zu diesem Modul gehörige ACL eine feingranuläre Einstellung der Bearbeitungsmöglichkeiten. So läßt sich ein Gruppenadministrator einrichten, der nur Benutzer einer bestimmten Gruppe bearbeiten darf und dabei auch nur bestimmte Attribute eines Logins verändern kann. Auch anhand der numerischen User- oder Gruppen-ID lassen sich solche Einstellungen durchführen.

15.9 cron

Neben dem at-Daemon ist der Cron-Daemon der wichtigste Dienst, mit dem sich auf einem Unix-System Programme zeitgesteuert starten lassen. Cron stützt sich dabei auf Tabellen (crontabs), die sowohl systemweit (»/etc/crontab«) als auch benutzerspezifisch (»/var/cron/tab«) definiert sein können. Während benutzereigene Crontabs ihre Prozesse immer unter der UID des jeweiligen Users starten, ist die Laufzeit-UID für Prozesse der systemweiten Crontab beliebig wählbar.

Für den Systemadministrator ist es entsprechend schwierig, hier die Übersicht zu den definierten Jobs zu behalten. Das Crontab-Modul der Gruppe *System* bietet hier eine Hilfestellung. Es zeigt alle Crontabs auf dem System in einer gemeinsamen Tabelle an. In einem HTML-Formular lassen sich Parameter wie Ausführungszeit und Benutzer-ID bestimmen.

Webmin zeigt dabei die systemweit definierten Cronjobs in einer Tabelle bei dem Benutzer an, unter dessen UID dieser Job läuft. Neue Einträge werden jedoch stets in einer benutzerspezifischen Crontab abgelegt. Einmal in einer bestimmen Tabelle definierte Jobs verbleiben dort auch bei einer Änderung.

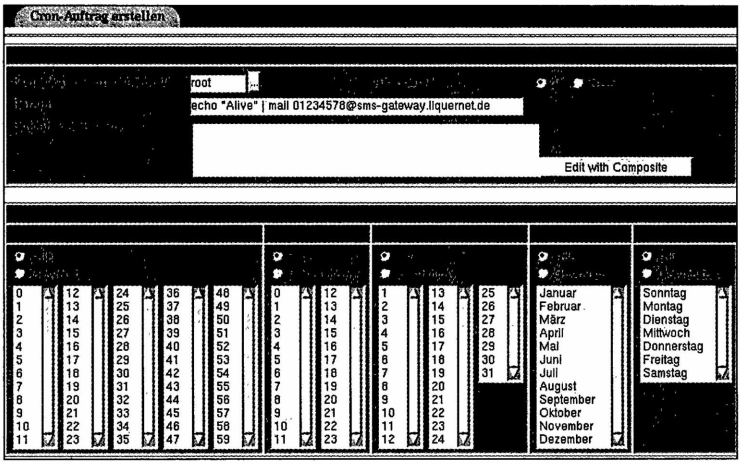


Abb. 15.8: Ein komfortables Eingabeformular für das Anlegen von Cronjobs

Leider verfügt das Modul nicht über die Möglichkeit, weitere Angaben neben den eigentlichen Jobdefinitionen in den Crontabs anzulegen. Dazu gehört unter anderem die Möglichkeit, einen Benutzer per E-Mail über Ausgaben und Fehlermeldungen seines Prozesses zu informieren. Diese Meldungen landen bei Fehlen einer benutzereigenen Einstellung immer im root-Postfach. Auch die Möglichkeiten der ACL dieses Moduls sind mager und beschränken sich darauf, die möglichen Benutzer-IDs für einen Prozeß einzuschränken. Immerhin läßt sich so vermeiden, daß ein Operator ein Skript über Cron mit Rootrechten ausführt. Trotz dieser Einschränkungen zeigt sich das Modul in der Praxis als hilfreich, da es ein direktes und fehlerträchtiges Editieren der Crontabs meist überflüssig macht und die verteilt gespeicherten Tabellen zentral anzeigt.

15.10 Die Softwareverwaltung

Nirgends ist die Heterogenität der Unix-Welt so greifbar wie bei der Installation von Softwarepaketen. Der System-V-Standard schreibt hier das Paketmanagement mit den Befehlen *pkgadd*, *pkginfo* und so weiter vor. Allerdings halten sich an diese Vorgabe von den großen Unix-Vertretern gerade einmal Solaris und SCO mit den Produkten OpenServer und UnixWare. HP-Unix geht mit den Tools *swinstall*, *swlist* und so weiter seine eigenen Wege bei der Verwaltung von Softwaredepots, und das Linux-Lager beschert uns mit der Debian-Paketverwaltung und dem RedHat-Package-Manager gleich zwei verschiedene Formate – einmal ganz abgesehen von der Slackware, die noch immer mit gepackten Tar-Archiven arbeitet.

Dieses Problem vor Augen ist die Softwareverwaltung innerhalb von Webmin besonders zu würdigen, da hier für den Reigen der Paketformate eine einheitliche Oberfläche bereitsteht. BSD-Pakete, System-V-Pakete, HP-UX-Softwaredepots, Debian-, RedHat- und Slackwarepakete können mit einer konsistenten Oberfläche installiert und deinstalliert werden. Bei NetBSD sind in einer Baumansicht alle unter »/usr/pkg« installierte

Pakete inklusive ihrer Paketbeschreibungen einsehbar. Jeder Paketname wird als Link dargestellt, hinter dem sich die Beschreibung und einer Liste aller Dateien des Pakets verbirgt. Auch ein Löschen eines Pakets ist von dieser Stelle aus möglich.

Die Installation neuer Pakete kann entweder von einer Datei auf der jeweiligen Maschine oder von einem FTP- oder HTTP-Server ausgehen. Auch der Upload einer Datei per HTTP ist möglich. Spezielle Optionen einzelner Paketformate sind dabei leider nicht zugänglich, hier verlangt die Universalität einige Abstriche. Ebenfalls bedauerlich ist das Fehlen weitergehender Konfigurationsmöglichkeiten oder einer differenzierten ACL.

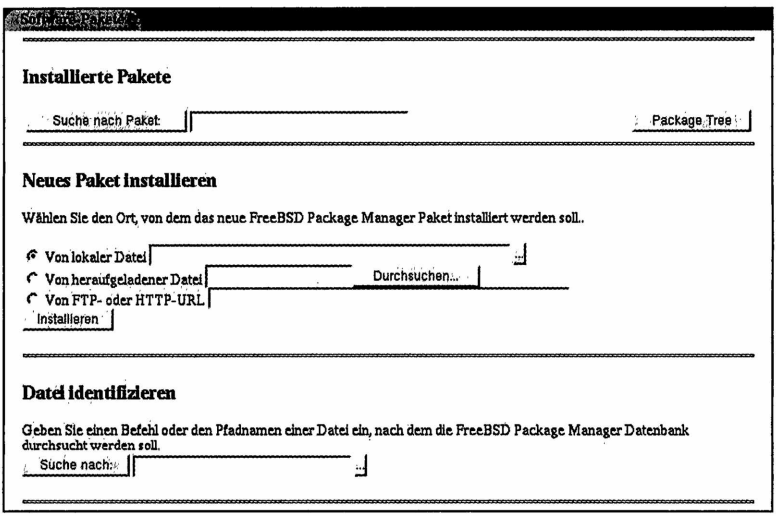


Abb. 15.9: Schnörkellos, aber funktionell und für viele Paketformate einheitlich – die Softwareverwaltung von Webmin

15.11 Start und Stop von Programmen

Die Systemgruppe von Webmin sammelt einige Module, die eine Kontrolle über die auf einem System laufenden Programme erlauben. Dazu gehören der Prozeßmanager, das Modul für den Start und Stopp von Daemonen beim Systemstart und bei System-V-konformen Unix-Betriebssystemen ein Inittab-Editor.

Um einen laufenden Prozeß zu beenden, öffnen Sie den Prozeß-Manager (*Laufende Prozesse*). Hier sehen Sie alle laufenden Prozesse, übersichtlich und nach Verwandtschaft geordnet. Das bedeutet, daß Tochterprozesse durch eine Einrückung ihrem Parent-Prozeß zugeordnet sind. Suchen Sie einen Prozeß aus und klicken Sie die PID (Prozeß-ID) an. Sie erhalten eine Maske, die Ihnen Informationen über den ausgewählten Prozeß mitteilt. Hier werden allgemeine Prozeßinformationen angezeigt, zum Beispiel auch der

Nice-Level eines Programms. Er bestimmt die CPU-Zeit, die einem Programm zur Verfügung gestellt wird (siehe Manpage zu *nice*). Im unteren Teil der Maske besteht auch die Möglichkeit, ein Signal an den Prozeß zu senden. Die am häufigsten genutzten Signale sind *KILL* und *TERM* zum Beenden eines Prozesses.

Das Modul *Systemstart/stopp* bietet eine Möglichkeit, das zentrale »/etc/rc.conf«-Skript zum Starten von Daemon-Prozessen zu editieren. Mehr als ein Telnet/vi-Ersatz ist das nicht, aber es erleichtert Routinearbeiten.

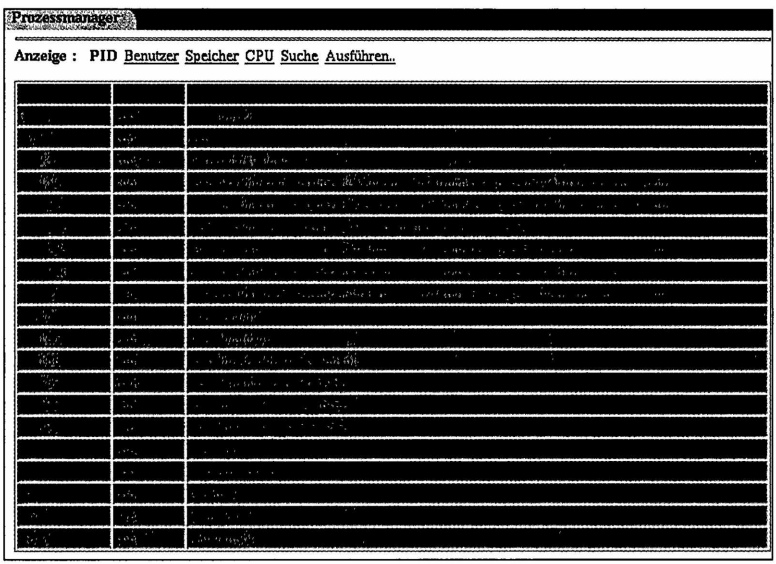


Abb. 15.10: Prozeßinformationen, wie sie Webmin auf einem einfachen NetBSD-System zeigt

15.12 Die Druckerverwaltung

Auch wenn manche EDV-Strategen noch immer gern vom papierlosen Büro schwärmen besteht die Hauptaufgabe vieler Rechner darin, am Ende bedrucktes Papier zu erzeugen. Entsprechend wichtig ist daher auch die Konfiguration des Drucksystems unter NetBSD. Dafür bietet Webmin eine entsprechende Hilfe in Form des *lpadmin*-Moduls (Druckerverwaltung) aus der Hardware-Gruppe an. Es funktioniert auf fast allen von Webmin unterstützten Plattformen und arbeitet sowohl mit dem BSD-lpd/lpr-System als auch mit *lprng* und dem neueren CUPS (Common Unix Printing System) zusammen.

Sofern neben Webmin auch Ghostscript installiert ist, können alle diesem Software-Raster-Image-Prozessor bekannten Druckertypen als Inputfilter für Druckerdefinitionen benutzt werden. Für NetBSD 1.6 bedeutet das die Installation des Pakets Ghostscript 6.01. Dafür sind neben dem Installationssets »xbase.tgz« (für die Library

»libXt.so.6«) die Pakete »freetype-lib 1.3.1«, »ghostscript fonts 6.0«, »watanabe-vfont« und »ja-vflib-lib« erforderlich. Entweder installiert man diese Pakete manuell vor der Installation von Ghostscript oder kopiert alle Pakete inklusive Ghostscript in ein Verzeichnis, um den Paketmanager bei der Ghostscript-Installation die vorhandenen Abhängigkeiten automatisch auflösen zu lassen. Das Ghostscript-Binary *gs* befindet sich wie üblich unter »/usr/pkg/bin«. Die Pfade zu diesem und anderen Programmen stehen in der Modulkonfiguration und sind leicht anzupassen.

Was ein Benutzer im einzelnen mit diesem Modul tun darf, kann über eine ACL ausreichend eingeschränkt werden. Nötig ist das freilich nur für Nicht-Postscriptdrucker beziehungsweise für Drucker, die an einem nicht postscriptfähigen Druckerserver hängen. Benötigt wird Ghostscript auch dann, wenn der NetBSD-Rechner selbst als ein solcher Server dienen soll.

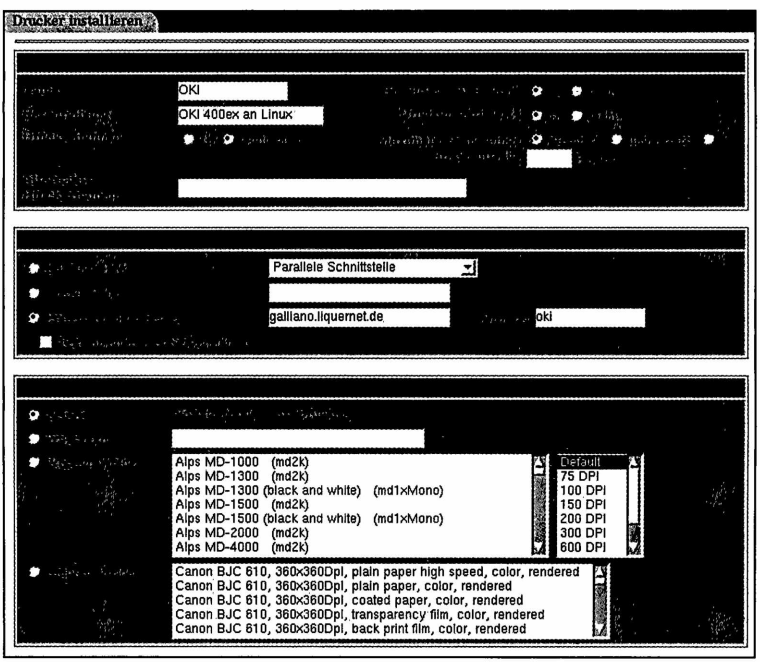


Abb. 15.11: Hier wird das Drucken auf einem lpd-Druckerserver definiert

Bei der Neueinrichtung eines Druckers werden dann alle Ghostscript bekannten Modelle tabellarisch angezeigt, die einzelnen Definitionen lassen sich in einer Maske editieren. Dabei können Drucker sowohl lokal als auch über das Netzwerk als Unix-lpd oder Windows-SMB-Shares erreichbar sein. Letzteres setzt eine lokale Installation des SMB-Servers *Samba* voraus, da für das Drucken auf einem Windowsdrucker das Programm *smbclient* aus dem Samba-Paket benötigt wird. Für jeden Drucker sind neben einigen

allgemeinen Einstellungen die Anschlußart und der zu verwendende Druckertreiber auszuwählen.

Anhand eines Beispiels soll verdeutlicht werden, welche Einstellungen Webmin an dieser Stelle am System vornimmt. Ziel soll es sein, einen Laserdrucker mit Namen *oki* von NetBSD aus zu benutzen, der an einem lpd-Druckerserver namens *galliano.liquernet.de* angeschlossen ist. Abb. 15.11 zeigt die notwendigen Einträge in der Webmin-Maske.

Zunächst werden diese Angaben zum Druckziel in die entsprechende Maske des Webmin-Moduls eingetragen und ein Name für diesen Drucker vergeben (hier *testoki*). Als Druckertreiber wird der Webmin-Treiber HP-Laserjet ausgewählt und eine passende Auflösung eingestellt.Ein Blick in die »/etc/printcap« bringt zutage, was Webmin mit diesen Angaben getan hat. Der neue Drucker findet sich unter dem definierten Namen eingetragen:

```
OKI|OKI 400ex an Linux:\
  sd=/var/spool/lpd/OKI:\
  rp=oki:\
  sh:\
  rm=galliano.liquernet.de
```

Webmin generiert also Einträge, wie sie der erfahrene Administrator sicher auch rasch formuliert hätte. So richtig spielt das Tool seine Fähigkeiten zusammen mit Samba und dem *smbclient* aus. Mit dieser Kombination druckt es sich problemlos auch auf Druckern an Windows-Rechnern.

Zunächst werden diese Angaben zum Druckziel in die entsprechende Maske des Webmin-Moduls eingetragen und ein Name für diesen Drucker vergeben (hier *smboki*). Als Druckertreiber wird der Webmintreiber *HP-Laserjet* ausgewählt und eine passende Auflösung eingestellt. Ein Blick in die »/etc/printcap« zeigt dann, was Webmin mit diesen Angaben getan hat. Der neue Drucker findet sich unter dem definierten Namen eingetragen:

```
smboki:sh:lp=/dev/null:\
:if=/etc/webmin/lpadmin/drivers/smboki.smb:\
:sd=/var/spool/lpd/smboki:
```

Webmin hat also einen Input-Filter (*if=*) für diesen Drucker eingetragen. Inputfilter werden vom hier verwendeten BSD-lpd/lpr-System verwendet, um Daten vor dem Druck aufzubereiten. Der hier eingetragene Filter *smboki.smb* umfaßt im wesentlichen zwei Schritte. Zunächst wird ein weiteres webmin-eigenes Skript *testoki* aufgerufen, dann werden die Daten mittels *smbclient* an den Windowsdrucker übergeben:

```
/etc/webmin/lpadmin/drivers/smboki "$1" "$2" "$3" "$4" "$5" "$6" "$7"
smbclient '\\galliano\oki' prlnt -U print \
-W liqueurnet -c "print /tmp/$$.smb" -P
```

Das Skript *testoki* schließlich ruft den Postscript-Interpreter Ghostscript auf, um die erhaltenen Daten für den gewählten Drucker aufzubereiten:

```
gs -sOutputFile=/tmp/$$.gs -dSAFER \
-sDEVICE=laserjet -r150 \
-dNOPAUSE /tmp/$$.gsi
```

Webmin besitzt also keine eigenen Druckertreiber, sondern erstellt aufgrund der eingetragenen Angaben für einen Drucker passende Inputfilter. Sofern Ghostscript und Samba auf einem Unix-Rechner installiert sind, ist das die beste Methode, um auch auf windows-basierten Netzwerkdruckern arbeiten zu können.

15.13 Internetdienste

In unserer kleinen Reise durch Webmin haben wir uns bisher wenig mit der Gruppe *Server* beschäftigt. Das soll sich nun ändern, hierfür betrachten wir folgenden Fall: Ein fiktives Unternehmen, die Foo GmbH, verfügt über eine Festverbindung zum Internet inklusive Firewallsystem. Neben dem reinen Internetzugriff für die Mitarbeiter entsteht der Wunsch, hinter dieser Internetanbindung auch einen eigenen Web- und Mailserver zu betreiben. Diesen Wunsch zu befriedigen, setzt zunächst die Bereitstellung eines hinreichend großen offiziellen IP-Netzes voraus, was über den Access Provider (sein Name sei »bar.de«) in der Regel problemlos möglich ist. Schließlich versorgt bereits ein Subnetz mit acht IP-Adressen (/29 in CIDR-Notation) neben Router und Firewall noch vier weitere Maschinen. Bei der Bereitstellung der Domain *foo-gmbh.de* sind die meisten Access-Provider ebenfalls gern behilflich und wickeln einen Großteil der Kontakte mit den zuständigen Registrars mit interaktiven Webtools ab. Für den geplagten Systemadministrator bleibt also die Einrichtung des Serversystems.

Die dazu notwendige Software bringt ein freies Unix-System wie NetBSD von Hause aus mit. Apache, Sendmail, BIND und ein FTP-Server sind entweder Bestandteil der Installationssets oder als Packages erhältlich. Die Konfiguration all dieser Serverkomponenten hat jedoch ihre Tücken. Webmin ereichtert die Arbeit dabei ungemein, da das Einarbeiten in die sehr unterschiedliche Syntax der verschiedenen Konfigurationsdateien weitestgehend entfällt. Wir werden in diesem Beispiel alle diese Serverkomponenten auf einer einzelnen Maschine in Betrieb nehmen, um ein möglichst einfaches Szenario zu erhalten. In der Praxis wäre dies keine gute Idee; zumindest Mail und DNS sollten auf einem, Web- und FTP-Server auf einem zweiten Rechner installiert werden. Websites nutzen oft externe Programme (CGI, PHP, JSP und so weiter), in denen Hintertüren und Trojaner verborgen sein könnten und FTP bietet viele interaktive Kommandos. Beide Funktionen passen nicht zu dem Anspruch, einen DNS- und Mail-Server mit maximaler Sicherheit und Verfügbarkeit aufbauen zu wollen.

15.13.1 Namensauflösung konfigurieren

Beginnen wir mit der Einrichtung eines DNS-Servers. Dazu sollte mindestens BIND8 herangezogen werden, da der Version 4 zahlreiche Zugriffskontrollmöglichkeiten fehlen. BIND in allen Versionen war und ist zudem ein beliebtes Ziel für Attacken, gegen die neuere BIND-Versionen besser geschützt sind. Die neueste Linie des Nameservers, BIND9, ist in dieser Hinsicht noch besser geschützt.

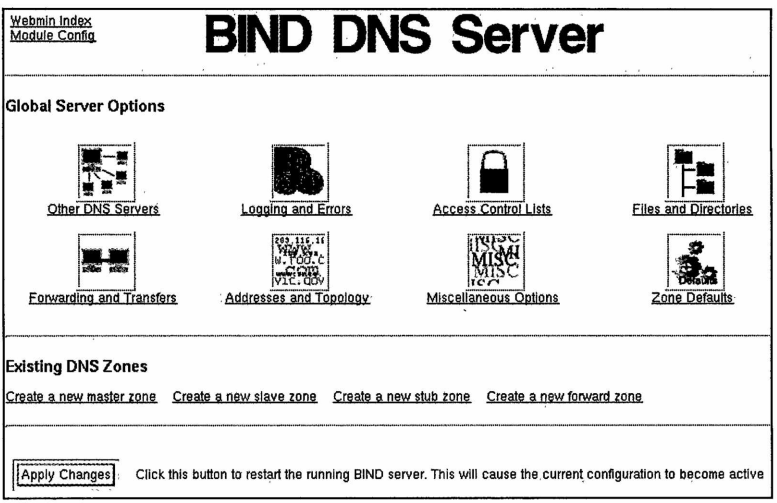


Abb. 15.12: So zeigt sich das DNS Modul für einen frisch installierten Nameserver

Der DNS-Server soll zunächst nur eine DNS-Zone verwalten: *foo-gmbh.de* enthält alle offiziellen Adressen, die vom Internet aus aufgelöst werden sollen. Dazu gehört der Haupt-Webserver *www.foo-gmbh.de*, ein weiterer Webserver *vertrieb.foo-gmbh.de*, der Mailserver *mail.foo-gmbh.de* und natürlich der DNS-Server selbst (*ns.foo-gmbh.de*). Beide Webserver erhalten dieselbe IP-Adresse (34.56.78.10) zugeordnet, denn wir werden die Unterscheidung dieser Webserver später auf Applikationsebene mit Apache durchführen. Die DNS-Zone ist eine primäre Zone, ihre Daten werden also direkt gepflegt und nicht von einem anderen Server repliziert. Der Access-Provider hat unserem Unternehmen ein Class-C-Netz 34.56.79.0 bereitgestellt und betreibt außerdem auf seinem DNS Server *ns.bar.de* (34.56.78.9) eine sekundäre Zone für *foo-gmbh.de*.

Sofern BIND frisch installiert wurde, wird der Aufruf des BIND-Moduls aus der Server-Gruppe das Fehlen der zentralen Konfigurationsdatei »/etc/named« anmahnen und eine neue Datei erzeugen wollen. Besteht zu diesem Zeitpunkt eine Internetverbindung, sollte man Webmin an dieser Stelle ein Update der Rootserver-Informationen gestatten. Veraltete Rootserver-Einträge können zu sporadisch auftretenden und daher schwer zu findenden Fehlern bei der Namensauflösung führen. Die erstellte Datei »/etc/named.conf« benutzt jedoch das »/etc«-Verzeichnis zur Speicherung der DNS-

Zonendateien. Will man, wie allgemein üblich, stattdessen die Zonenfiles unter »/var/named« ablegen, kommt man um ein manuelles Edieren der Datei nicht herum.

Nachdem dieser Vorgang abgeschlossen ist und der Nameserver gestartet wurde, sollte sich das BIND-Modul, wie in Abb. 15.12 gezeigt, darstellen. Bevor wir unsere erste Zonendatei anlegen, sind einige Vorarbeiten notwendig. In der *Modul Konfiguration* sollte der *Serial Number Style* auf *date based* eingestellt werden, um anhand der Seriennummer einer Zonendatei leichter auf das Alter der Einträge schließen zu können. Unter *Forwarding and Transfers* tragen wir den Nameserver des Providers als Forwarder ein, unter *Zone defaults* erlauben wir außerdem für diesen Server den Zonentransfer als Vorgabe für alle Zonen (Abb. 15.13). Damit kann die Erstellung der DNS-Zonen beginnen.

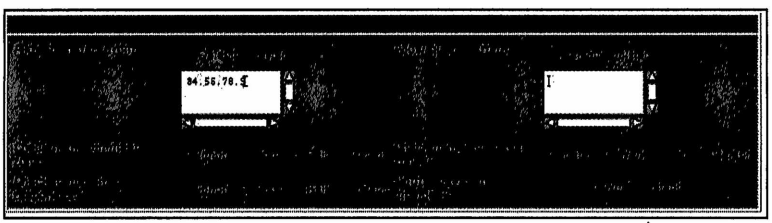


Abb. 15.13: Die Grundeigenschaften einer DNS Zone sind mit Webmin rasch eingestellt

Über den Link *Create a new master zone* gelangen wir in eine Maske, die uns die grundliegenden Angaben für unsere Zonen abfordert. Die erste Zone, die wir einrichten, ist eine »Reverse«-Zone für die Auflösung von IP-Adressen zu Rechnernamen. Die Adresse des Netzwerks ist ohne Netzmaske anzugeben, ein Class-C-Netz ist zum Beispiel als 34.56.78 einzutragen. Außerdem sind die Angaben zum Master-Server selbst und die Mail-Adresse des Serververwalters anzugeben, die übrigen Vorgaben können wir übernehmen.

Anschließend erstellen wir eine »Forward«-Zone für die Domain *foo-gmbh.de*. Darin müssen über die Maske *Adresses* die Namen und IP-Adressen der Hosts eingetragen werden. Es genügt der Hostname, der Domainanteil ergibt sich automatisch aus dem Zonennamen. Webmin nimmt uns die Pflege der »Reverse«-Zone ab und trägt jede IP-Adresse mit dem dazugehörigen Rechnernamen in die »Reverse«-Zone ein. Außerdem ergänzen wir unter *Name servers* den Provider-DNS *ns.bar.de*, der ja als Sekundär-DNS dienen soll und tragen unter *Mail servers* unseren Mailserver ein.

15.13.2 Sicherheitsaspekte

Besonderes Augenmerk ist nun auf die *Zone options* zu richten. Damit der Secondary DNS des Providers einen Zonentransfer für die eben eingerichtete Zone durchführen kann, ist die IP-Adresse seines DNS in die Liste *Allow transfers from* aufzunehmen. Das verhindert außerdem ein »Ausspionieren« der Einträge in der Zone, da auch das nslookup-Kommando *ls* einen Zonentransfer vollführt. Wer also die Hostnamen nicht kennt, erhält auch keine Namensauflösung – ein Stein mehr im Weg eines Angreifers. Wir

könnten in der Liste *Allow queries from* auch die normale DNS-Abfrage einschränken und nur bestimmten Netzen Zugriff auf die Zone erlauben. Für eine Zone, die öffentlich erreichbare Server auflöst, ist dies wenig sinnvoll, für Zonen mit privaten Adressen aus einem LAN hingegen schon. Diese Adressen gehen niemanden außerhalb des Unternehmens etwas an und können hier vor fremdem Zugriff geschützt werden.

Webmin bietet übrigens über die zum BIND-Modul gehörende ACL eine brauchbare Möglichkeit, die Verwaltung des DNS zu delegieren. Dazu legt man wie üblich zunächst einen passenden Webmin-Benutzer (*hostmaster*) an und erlaubt diesem Benutzer nur den Zugang zum BIND-Modul. Durch die Einstellungen der ACL könnten dem Hostmaster nun zum Beispiel alle Rechte bis auf den Neustart des BIND-Servers gegeben werden. Erst der Haupt-Administrator würde dann nach einer Kontrolle der vorgenommenen Änderungen diese freigeben, ein »Vier-Augen-Prinzip« bliebe also gewahrt.

Nach einem Neustart des DNS Servers können wir unseren DNS-Server testen, dazu dient das Kommando *nslookup* aus einer Shell heraus. Die Eingabe von *www.foo-gmbh.de* sollte die korrekte IP-Adresse liefern. Nach einem *set q=MX* können wir durch Eingabe des Domainnamens auch den MX-Record durch Eingabe des Domainnamens prüfen, ein *set q=SOA* zeigt in gleicher Weise die Grundeinstellungen zu der jeweiligen Zone an. Funktioniert alles einwandfrei, sollte der Betreiber des Seconday DNS informiert werden, damit er eine »Salve Zone« einrichtet und unsere Einträge übernimmt.

15.13.3 Sendmail konfigurieren

Für viele Systemadministratoren gehört Sendmail zu den am schwierigsten zu beherrschenden Programmen und besonders ein Blick in den hinteren Teil der Konfigurationsdatei »/etc/sendmail.cf« trägt zur Festigung dieses Eindrucks entscheidend bei. Auch in Bezug auf die Systemsicherheit steht Sendmail bei vielen nicht im besten Ruf, war dieser Mail Transfer Agent oft Ziel von Angriffen. Mit der unlängst erschienenen Version 8.11 ist zumindest das letzte Argument hinfällig, da diese Sendmail viele Sicherheits-Features bietet. Außerdem ist das 1981 von Eric Allman begonnene Sendmail-Projekt nicht nur eines der ältesten Open-Source-Projekte, sondern auch eines der am besten getesteten Software-Produkte überhaupt. Auch die Konfiguration läßt sich entscheidend erleichtern – entweder durch das kommerzielle Tool von Allmans Firma Sendmail, Inc. oder durch das entsprechende Webmin-Modul.

Das Modul findet sich unter *Servers, Sendmail Configuration* und bietet sowohl Zugang zu allgemeinen Parametern der Sendmail-Konfigurationsdatei als auch zu den diversen Tabellen, aus denen Sendmail Informationen zieht. Der besseren Übersichtlichkeit halber seien diese Tabellen hier kurz beschrieben:

- Die Mail-Alias-Tabelle (»/etc/aliases«) ordnet lokalen Benutzern Aliasnamen zu. Solange die Zielbenutzer als Benutzer auf der lokalen Maschine eingetragen sind, lassen sich so auch einfache Gruppen (Mailing Lists) erstellen. Über diese Tabelle können Mails auch über einen Pipe-Mechanismus an ein externes Programm weitergeleitet werden.

- Die *Local Domain*-Tabelle (»/etc/mail/sendmail.cw«) listet die Domains auf, die der lokale Mailserver selbst verwaltet. Im einfachsten Fall ist dies der Hostname der Maschine selbst, in unserem Beispiel zusätzlich auch der Name der Firmendomain (*foo-gmbh.de*).
- Ähnlich wie die Alias-Tabelle dient auch die *Virtual User*-Liste (»/etc/mail/virtusertable«) zur Umleitung von Mails an andere Adressen, jedoch kann die Zieladresse dabei in einer beliebigen Domain liegen. Wir werden diese Tabelle im weiteren noch etwas eingehender betrachten.
- Zum Kampf gegen die Ölpest des Internets, die als Spam bezeichneten Werbe- und Unsinnmails, bietet Sendmail zwei geeignete Werkzeuge, nämlich die Spam-Control- und Relaying-Tabellen. Mittels *Spam control* (»/etc/mail/access«) lassen sich anhand von Absenderadresse oder IP-Adresse beziehungsweise der Domain des sendenden Mailservers Regeln für die Annahme oder Ablehnung von Mails definieren und so der Empfang von Spam-Mails verhindern. Umgekehrt regelt die *Relay Domains*-Tabelle, an welche Domains der Server Mails weiterleiten darf. Dadurch wird dem Mißbrauch des eigenen Servers durch Spammer zum Versenden von digitalen Belästigungen Einhalt geboten. Diese Maßnahmen liegen im ureigenen Interesse des Systemadministrators, denn ein offenes Mailrelay kann schlimme Konsequenzen haben. Organisationen wie ORBS [3] führen nämlich Listen mit offenen Mail-Relays und auch Sendmail läßt sich so konfigurieren, daß Mails von Servern auf einer solchen Liste generell abgelehnt werden. Ein so erfaßter Server wird also E-Mails an viele Domains nicht mehr versenden können.

Des schnelleren Zugriffs wegen benutzt Sendmail übrigens nicht die Tabellen in ihrer ASCII-Form, sondern stützt sich auf Hashtabellen oder das Berkeley-DBM-Format. Die Generierung der entsprechenden Datendateien nimmt Webmin nach jeder Änderung automatisch vor. Die Änderungen wirken sich sofort auf die laufende Sendmail-Instanz aus, ein Neustart ist nicht erforderlich.

15.13.4 Virtuelle Benutzer

In unserem Beispiel begnügen wir uns mit einer einfachen Konfiguration, indem wir den Mailserver auch als POP3-Server benutzen, alle Postfächer also auf derselben Maschine führen. Zunächst wird als lokale Domain neben dem Server selbst (*mail.foo-gmbh.de*) nur *foo-gmbh.de* eingetragen. Für jeden Benutzer richten wir dann über die Benutzerverwaltung von Webmin einen Unix-Benutzeraccount ein. Diese Accounts ermöglichen das Abrufen erhaltener Mails über den meist standardmäßig aktiven POP3-Daemon. Jeder Benutzer erhält als Login-Shell */bin/false* zugeordnet, wodurch ein Login an der Konsole oder über Telnet verhindert ist. Zur Unterscheidung von echten Benutzern numerieren wir diese Benutzer einfach fortlaufend in der Art *pop001*, *pop002* und so weiter, die tatsächlichen E-Mail-Adressen wie *Rincewind@foo-gmbh.de* bilden wir (Abb. 15.14) als virtuelle Benutzer ab. Auf den ersten Blick ist dies ein Mehraufwand, zahlt sich aber durch mehr Flexibilität bei der Vergabe von Mail-Adressen aus. Außerdem sind auch Mailumleitungen auf externe Adressen auf diese Weise einfach möglich.

Zur Sicherung des Servers vor Mißbrauch tragen wir das verwendete offizielle Netz (34.56.78.0) sowie das im LAN benutzte RFC1918-Netz (hier 192.168.4.0) in die entsprechende Tabelle ein. Dadurch können, entsprechende Firewallregeln vorausgesetzt, alle lokalen Arbeitsplätze den Mail-Server als SMTP-Server für ausgehende Mails benutzen. Das hat seine Vorteile, denn wenn ein Ziel-Mailserver einmal für eine gewisse Zeit nicht erreichbar ist, bleibt das vom Arbeitsplatz unbemerkt. Der SMTP-Server behält die Mail in seiner Warteschlange und stellt sie zu, sobald der Partner wieder erreichbar ist.

Das Sendmail-Modul von Webmin bietet noch viele weitere Möglichkeiten als hier geschildert, doch für die hier zugrunde gelegte Problemstellung soll die Betrachtung genügen.

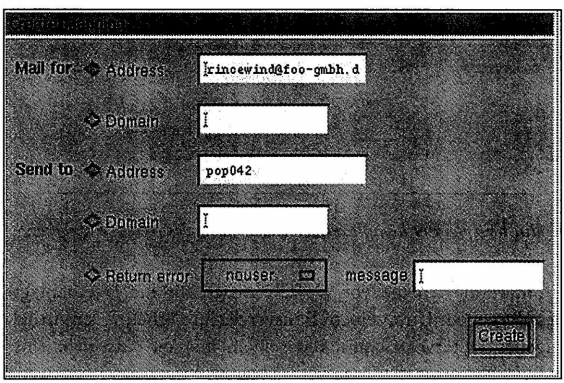


Abb. 15.14: Mailadressen lassen sich über die virtusertable von Sendmail auf Betriebssystemaccounts abbilden

15.13.5 Das Apache-Modul

Ursprünglich als Sammlung von Patches für den CERN httpd begonnen, ist Apache mittlerweile eines der erfolgreichsten Open-Source-Projekte. Gemessen an der Anzahl betriebener Websites ist er der führende Webserver weltweit und bietet durch seine modulare Struktur vielfältige Anpassungsmöglichkeiten. Entsprechend komplex ist auch seine Konfiguration, wenngleich nicht so verwirrend in der Syntax wie Sendmail. Das Webmin-Modul für Apache ist gut geeignet, der Apache-Konfiguration die letzten Schrecken zu nehmen und besonders Anhängern des Pendants aus Redmont den Umstieg zum Marktführer schmackhaft zu machen.

Unsere Foo GmbH benötigt zwei getrennte Webpräsenzen: eine für die allgemeine Firmenpräsentation im Internet (*www.foo-gmbh.de*) und eine zweite mit Informationen für Vertriebsmitarbeiter im Außendienst (*vertrieb.foo-gmbh.de*). Während die erste Site allgemein zugänglich sein soll, erfordert die zweite Site Benutzernamen und Paßwort für den Zugriff auf die abgelegten Daten. Für die Einrichtung beider Sites benutzen wir den

Mechanismus der *Virtual Servers* von Webmin. Er erlaubt den Betrieb vieler logischer Sites auf einem physikalischen Server. Die Unterscheidung der Sites kann auf zwei Arten erfolgen: Adreßbasierte, virtuelle Server werden an unterschiedlichen IP-Adressen erkannt, namensbasierte hingegen am Namen des URLs. In beiden Fällen ordnet Apache dem virtuellen Server ein bestimmtes Dokumentenverzeichnis zu. Beim adreßbasierten Mechanismus sind jedoch mehrere offizielle IP-Adressen auf Betriebssystemebene zu vergeben, der namensbasierte Mechanismus begnügt sich mit einer einzigen. Um IP-Adressen zu sparen, entscheiden wir uns hier für namensbasierte, virtuelle Server. Die Vorarbeit haben wir bei der DNS-Einrichtung getan, indem wir sowohl *www.foo-gmbh.de* als auch *vertrieb.foo-gmbh.de* dieselbe IP-Adresse zugeordnet haben.

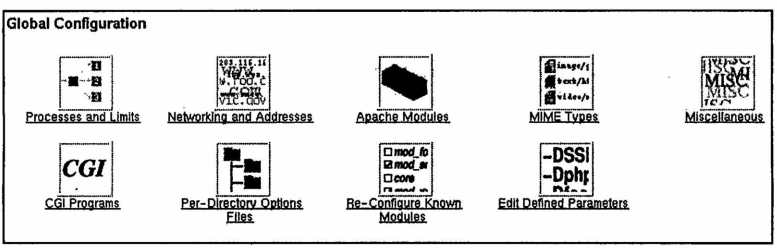


Abb. 15.15: Das Apache-Modul hilft bei der komplexen Konfiguration des freien Webservers

Etwas Vorüberlegung erfordert die Anlage von Verzeichnissen für die Datenablage. Bewährt hat sich folgendes Vorgehen: Unter einem Basisverzeichnis erhält jeder virtuelle Server zunächst ein eigenes Unterverzeichnis, das seinem Namen entspricht (hier »www« und »vertrieb«). Darin legen wir die Unterverzeichnisse »log«, »etc«, »site« und »site/htdocs« an. Dieser gesamte Verzeichnisbaum gehört der Unix-Benutzergruppe, unter der unser Webserver läuft (hier *apache*), sie hat Leserechte auf den gesamten Baum und Schreibrechte im Verzeichnis »logs«.

Nach dem Wechsel in das Apache-Webmin-Modul bietet sich die in Abb. 15.15 dargestellte Ansicht. Zunächst müssen wir festlegen, welcher offiziellen IP-Adresse wir namensbasierte Server zuordnen wollen. Das geschieht in der Maske *Networking and Adresses* in der Liste *Adresses for name virtual hosts*; hier ist die IP-Adresse 34.56.78.10 einzutragen. Mit der Dialogbox *Create new virtual server* (Abb. 15.16) erzeugen wir nun die beiden Server. Die Adresse ist in beiden Fällen 34.56.78.10, der Servername *www.foo-gmbh.de* beziehungsweise *vertrieb.foo-gmbh.de*, Document-Root jeweils das »htdocs«-Verzeichnis der oben erzeugten Benutzer.

Alle so erzeugten virtuellen Server erhalten in der Startseite des Apache-Moduls ein eigenes Icon, über das weitere Konfigurationen möglich sind. Dabei wirken übrigens Einstellungen am Defaultserver global für alle virtuellen Server, sofern dort nicht andere Parameter bestimmt wurden. Wenden wir uns zunächst der Protokollierung von Seitenabrufen zu. Hinter dem Icon *Log files* eines virtuellen Servers stellen wir das Access-Log auf die Datei »logs/access.log« in der Verzeichnisstruktur ein. So werden Zugriffe auf die beiden virtuellen Server getrennt protokolliert und lassen sich auch separat auswerten, zum Beispiel mit dem Webalizer [2]. Für Fehlermeldungen hinge-

gen sollte man eine gemeinsame Logdatei benutzen, da sich Zugriffsprobleme und so weiter zentral leichter erkennen lassen.

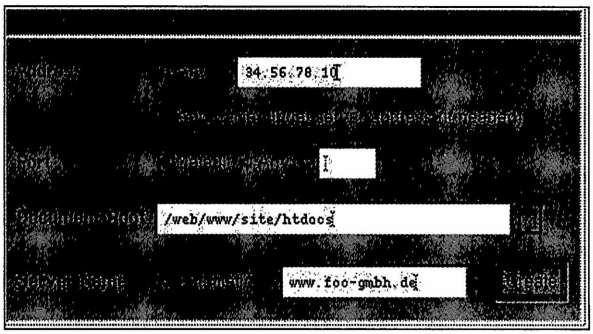


Abb. 15.16: Mehrere Domains auf einem Webserver werden als virtuelle Server konfiguriert und betrieben

15.13.6 Zugangskontrolle

Der Server für den Vertrieb ist in der gegenwärtigen Weise noch für das gesamte Internet zugänglich. Für die gewünschte Zugriffsbeschränkung verlangt Apache die Definition eines Verzeichnisses, denn die Einrichtung von Access Control Lists wird niemals auf Serverebene, sondern immer auf Verzeichnisebene vorgenommen. Das zu schützende Verzeichnis ist in unserem Falle das Document-Rootverzeichnis selbst (»/www/vertrieb/site/htdocs«). Für dieses Verzeichnis wird mit der Dialogbox *Create Per-Directory, Files or Location Options* ein eigenes Symbol erzeugt und die Funktion *Access control* hinter diesem Icon aufgerufen. In der Dialogbox setzen wir den *Authentication type* auf *Basic*, stellen *Restrict access by login* auf *All valid users* und definieren die Datei »/web/vertrieb/etc/acl« als *User Text File*. Darin werden ähnlich der »/etc/passwd« des Betriebssystems die Benutzernamen und verschlüsselte Paßwörter abgelegt. Nach dem Speichern der Angaben erscheint neben der Pfadangabe ein Link *Edit users*, über den wir neue Benutzernamen definieren können. Das Paßwort kann im Klartext oder auch bereits verschlüsselt aus der »/etc/passwd« eingegeben werden, wenn der angelegte Benutzer schon auf Betriebssystemebene angelegt ist.

Übrigens erlaubt die ACL des Apache-Moduls die Beschränkung der Möglichkeiten eines Webmin-Benutzers auf bestimmte virtuelle Server und bestimmte Bereiche der Konfiguration. Wir könnten die Pflege der Benutzerliste daher an die Vertriebsabteilung delegieren, indem wir einen Webmin-Benutzer *webmaster-vertrieb* erzeugen und diesem lediglich Zugang zu seinem virtuellen Webserver gewähren.

Da Apache bei jedem Login auf ein geschütztes Verzeichnis direkt in die jeweilige ACL-Datei zugreift, erfordern Änderungen an dieser Datei keinen Neustart des Webservers – somit benötigt unser Webmaster dieses Recht nicht.

Jetzt ist es an der Zeit, mit dem Link *Apply changes* aus der Startseite des Moduls den Apache-Server zum Einlesen der geänderten Konfiguration zu bewegen. Unsere virtuellen Webserver sind damit einsatzbereit. Natürlich läßt die Konfiguration in dieser Weise noch viele Wünsche offen – beispielsweise fehlt unter anderem ein Verzeichnis für CGI-Programme, der Umgang mit leeren Verzeichnissen (Indexing) ist nicht geregelt und dergleichen mehr. Ein Blick in die Dokumentation zu Apache ist hier sehr hilfreich, um die Prinzipien der Arbeitsweise kennenzulernen. Bei der syntaktisch korrekten Umsetzung ist Webmin dann eine große Hilfe.

15.13.7 Ablage der Webdaten

Was den Webservern nun noch fehlt, sind Daten in Form von HTML- und Grafikdateien. Kaum ein Unternehmen begnügt sich dabei heute auf einige selbstgestrickte Webseiten wie zu Beginn der Internet-Ära, vielmehr wird versucht, die Corporate Identity durch Werbefachleute auch ins Internet zu portieren. Was auch immer man von diesen Sites halten mag, wenigsten fördern sie die Nachfrage an Breitbandtechnologien – doch das ist ein anderes Thema. Offen ist für unsere Systeme hier die Frage, wie die erstellten Seiten auf die Serverplatte gelangen sollen. Auch wenn sich durch Protokolle wie WebDAV elegante Möglichkeiten eröffnet haben, ist der klassische Weg noch immer das *File Transfer Protocol* FTP. Hier ist Vorsicht geboten: Nach einem erfolgreichen Login ist auch der Weg zu Systemdateien wie der »/etc/passwd« und anderen sicherheitsrelevanten Dateien offen. Etwas Mißtrauen schützt hier vor bösen Überraschungen.

Der verbreitete FTP Server *WU-FTPD*, dessen Wurzeln bis in die achtziger Jahre zurückreichen, bietet durch die Integration des unix-spezifischen »Change Root«-Mechanismus ein recht zuverlässiges Mittel. Dieser FTP-Server wird standardmäßig durch Webmin unterstützt; Anhänger anderer Server wie des *ProFTPD* sollten einen Blick auf die Third Party Modules [1] werfen.

15.13.8 WU-FTPD administrieren

Bestandteil der *wuftpd*-Distribution, meist jedoch als eigenes Paket beigefügt, ist eine Musterumgebung für einen anonymen FTP-Benutzer, auf die wir zur Sicherung unserer Webverzeichnisse zurückgreifen. Es handelt sich hierbei um eine vorbereitete *chroot*-Umgebung, die wir unter Erhaltung aller Zugriffsrechte (*cp -a*) in die *site*-Verzeichnisse der virtuellen Webserver kopieren. Für jeden Webserver legen wir nun einen eigenen Benutzer an, der Mitglied in der Gruppe *apache* beziehungsweise der Gruppe wird, unter der der Apache-Webserver läuft. Als Heimatverzeichnis wählen wir das *site*-Verzeichnis aus. Anschließend bestimmen wir den angelegten Benutzer als Eigentümer des Verzeichnisses »htdocs« und gewähren ihm Schreibrechte auf dieses Verzeichnis. Damit sind die Vorarbeiten abgeschlossen und wir können uns dem FTP-Server zuwenden.

Die Hauptauswahl des WU-FTPD-Moduls ist in Abb. 15.17 gezeigt. Hinter dem Icon *Users and classes* lassen sich verschiedene Einstellungen für den Umgang mit User-Logins vornehmen. Im Feld *Unix groups and GIDs to treat as guests* tragen wir die Gruppe *apache* ein. Gast-Logins haben zwar in der Regel alle Privilegien eines normalen Benutzers, werden aber durch den oben erwähnten *chroot*-Mechanismus auf ihr Heimatverzeichnis beschränkt, ein Wechsel in eine höhere Verzeichnisebene ist nicht möglich. Anschlie-

ßend stellen wir unter *Miscellaneous options* die *Default umask for uploaded files* auf 644, um die korrekten Schreib-/Leserechte für Benutzer und Gruppe bei neuen Dateien zu erhalten.

Nach einer Anmeldung mit einem FTP-Client sollte man direkt in das entsprechenden *site*-Verzeichnis des Webservers gelangen und nicht in höhere Ebenen wechseln können.

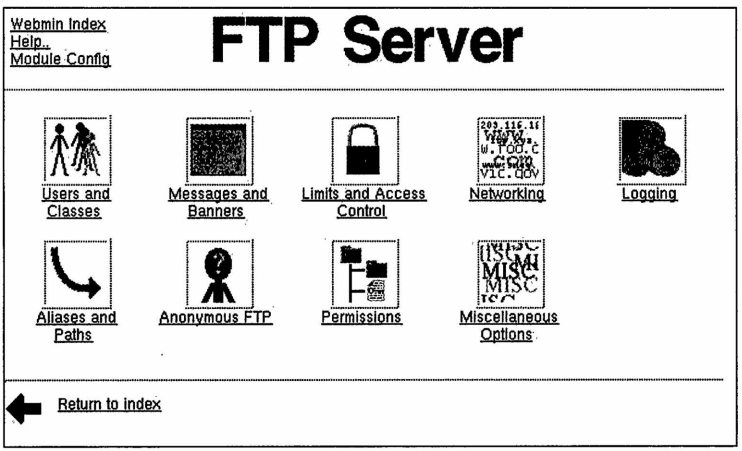


Abb. 15.17: Das Einrichten des WU-FTPD

Fazit

Die Einrichtung eines firmeneigenen Internet-Serverparks ist vollständig mit freier Software möglich. Dabei sind in der Leistungsfähigkeit keinerlei Abstriche in Kauf zu nehmen, vielmehr zählen Sendmail und Apache zu den am häufigsten eingesetzten Programmen für den jeweiligen Zweck und können sich mit Fug und Recht als Marktführer bezeichnen. Das kommt nicht von ungefähr, da sie eine Vielzahl von Möglichkeiten bieten und die kommerzielle Konkurrenz oft übertreffen. Webmin kommt in einer solchen Umgebung die wichtige Rolle einer zentralen Administrationshilfe zu, die ein direktes Eingreifen in die jeweiligen Konfigurationsdateien entbehrlich macht. Ein auf diese Art gepflegter NetBSD-Server ist nicht aufwendiger in der Administration als ein beispielsweise mit Windows eingerichteter Server, arbeitet jedoch erheblich stabiler.

Literaturverweise

[1] http://webmin.thridpartymodules.com

[2] http://www.webalizer.com

[3] http://www.abuse.net

16 Der NetBSD-Sourcetree

von Jörg Braun

Wie bei freier Software üblich, wird zu NetBSD der komplette Quelltext des Betriebssystems ausgeliefert. Man benötigt die Quellen allein schon dann, wenn man eine Aktualisierung vornehmen will oder einen neuen Kernel baut. Mit den Quellen läßt sich aber mehr machen: Sie dienen dem Aufbau eines Distributionssets inklusive der Installations-Diskettenimages, man kann damit auch die Quellen komplett für eine andere Plattform kompilieren (Cross Compile) oder auch nur nachsehen, welche Treiber von wem wie programmiert wurden. Was nicht zum originären Sourcetree gehört, sind die externen Programme des Package-Verzeichnisses und die Quelltexte von X.

In dieser Beschreibung geht es ausschließlich um den Aufbau des Standard-Quelltextverzeichnisses. Es wird in mehreren »tgz«-Dateien zur Verfügung gestellt und wird mit dem *tar*-Befehl eingebunden.

Datei	Daten	Archivgröße	Ausgepackt
gnusrc.tgz	Quelltexte unter GNU-Lizenz	ca. 55 Mbyte	ca. 250 MByte
sharesrc.tgz	Allgemein benötigte Quellen	ca. 3,5 Mbyte	ca. 15 MByte
src.tgz	Quelltexte und BSD-Lizenz	ca. 26,5 Mbyte	ca. 135 MByte
syssrc.tgz	Kernel-Quellen	ca. 22 Mbyte	ca. 115 MByte

Tabelle 16.1: Die Quelltext-Archive (Größen unter NetBSD 1.6)

Ausgepackt werden müssen die Quellarchive aus dem Hauptverzeichnis, dazu müssen *root*-Rechte vorhanden sein:

```
[/]#tar -xzf /mnt/cdrom/source/sets/gnusrc.tgz
[/]#tar -xzf /mnt/cdrom/source/sets/src.tgz
[/]#tar -xzf /mnt/cdrom/source/sets/srcsrc.tgz
[/]#tar -xzf /mnt/cdrom/source/sets/syssrc.tgz
```

Anschließend befinden sie sich in einigen neuen Unterverzeichnissen von »/usr/src«. Es ist also unbedingt darauf zu achten, daß entweder die »/usr«-Partition groß genug ist oder eine zusätzliche Partition nach »/usr/src« eingebunden wird.

Die Verzeichnisse, die unter »/usr/src« nach dem Einspielen der Archive angelegt sind, enthalten jeweils relativ voneinander unabhängige Verzeichnisbäume. Die Verzeichnisnamen korrespondieren mehr oder minder mit den Namen der Binärverzeichnisse:

Verzeichnis	Inhalt
bin	Hier enthalten sind die Quellen der Programme in »/bin«.
crypto	Verschlüsselung. Wird von den anderen Verzeichnissen benötigt.
dist	Externe Quellen, die zum Baum gelinkt werden (»distributed«).
distrib	Die Distributionsdateien (Floppy-Images und dergleichen).
etc	Hier enthalten sind die Dateien, die später nach »/etc« gehören.
games	Die BSD-Spiele. Die Programme befinden sich in »/usr/games«.
gnu	Die Dateien aus »gnussrc.tgz«. Hier sind dann wieder die Unterverzeichnisse »dist«, »lib«, »libexec«, »usr.bin« und »usr.sbin« zu finden.
include	Die Include-Dateien, nach der Installation in »/usr/include«.
lib	Bibliotheksdateien für »/usr/lib».
libexec	Daemons und Systemdateien für »/usr/libexec«.
regress	Regressionstests.
sbin	Systemprogramme in »/sbin«.
share	Gemeinsam genutzte Dateien für »/usr/share«.
sys	Die Kernel-Quellen.
tools	Build-Tools für den Sourcetree.
usr.bin	Die Sourcen der Programme in »/usr/bin«.
usr.sbin	Die Sourcen der Programme in »/usr/sbin«.

Tabelle 16.2: Die Unterverzeichnisse des Quelltextbaums von NetBSD

Das Verzeichnis »/usr/src« selbst enthält einige Dateien, die für das Verständnis und die Arbeit mit dem Quelltextsystem von Bedeutung sind:

Datei(en)	Funktion
BUILDING/BUILDING.mdoc	Dokumentation zum System-Build.
UPDATING	Beschreibung, wie man das System aktualisiert.
Makefile/Makefile.inc	Die Make-Datei (und Include) für den Build-Vorgang.
build.sh	Das Skript für die weitgehende Automatisierung des Builds.

Tabelle 16.3: Die Dateien in »/usr/src«

Die Dokumentationen sollten vor umfangreicheren Arbeiten unbedingt durchgelesen werden.

Wie man einen angepaßten Kernel baut, soll an dieser Stelle nicht diskutiert werden. In diesem Kapitel geht es darum, wie man Teile des Quelltextbaums oder auch den kompletten Tree kompiliert und wie man eine Distribution für eine andere Maschine erzeugt (Cross Compile).

16.1 Das Kompilieren von Verzeichnissen

Jeder Teil des Quelltextbaums von NetBSD – Ausnahmen bestätigen die Regel – enthält eine Make-Datei. Durch den Aufruf des Programms *make* wird diese Datei verarbeitet und der aktuelle Bereich des Baums rekursiv kompiliert. Das bedeutet, daß alle vorhandenen Unterverzeichnisse bearbeitet werden. Es werden aber beim selektiven Übersetzen eines Teils des Baums Abhängigkeiten aus anderen Teilen des Sourcetrees nicht aufgelöst. Bei Bedarf, das heißt, weil wegen einer fehlenden Datei abgebrochen wird, muß dieser Bereich ebenfalls übersetzt werden. Erst dann kann an der aktuellen Stelle fortgefahren werden. Das kann problematisch sein, wenn man nicht weiß, wo die bemängelten Dateien sind.

Nehmen wir einfach an, es soll der komplette Sourcetree so übersetzt werden, daß inklusive der Disketten-Images und der Distributions-Sets alle Daten erzeugt werden. Um diese Arbeit manuell durchzuführen, müssen im ersten Schritt alle Installations-Kerneldateien erzeugt werden. Bei NetBSD 1.6 sind das recht viele: »INSTALL«, »INSTALL_SMALL«, »INSTALL_TINY«, »INSTALL_PS2« und »INSTALL_LAPTOP«. Außerdem schadet es keineswegs, auch die beiden Kernel-Dateien »GENERIC« und »GENERIC_LAPTOP« zu erzeugen. Sie werden für ein Installationsset ebenfalls benötigt.

Um diese Arbeiten »halbautomatisch« zu erledigen, muß in das Verzeichnis »/usr/src/sys/arch/i386/conf« gewechselt werden. Hier müssen die entsprechenden Kernel-Konfigurationdateien mit *config* bearbeitet werden:

```
# for a in GENERIC* ; do config ${a} ; done
# for a in INSTALL* ; do config ${a} ; done
```

In diesem Schritt werden die Verzeichnisse für den Kompiliervorgang angelegt. Nun muß nach »../compile« gewechselt werden. Hier müssen die Kernel-Dateien selbst gebaut werden:

```
# cd ../compile
# for a in INSTALL* ; do cd ${a} ; make depend && make ; cd .. ; done
# for a in GENERIC* ; do cd ${a} ; make depend && make ; cd .. ; done
```

Hierbei wird in alle angelegten Verzeichnisse gewechselt. Dann werden die Abhängigkeitslisten aufgebaut und die Kernel-Dateien, die alle den Namen »netbsd« haben, kompiliert.

Da es sich um eine ganz erkleckliche Zahl von Kernel-Dateien handelt, dauert das eine geraume Zeit.

Im nächsten Schritt müssen die Installations-Bootfloppies gebaut werden, das ist recht einfach, wenn die Kernel-Dateien vorhanden sind. Man wechselt nach »/usr/src/distrib/i386/floppies« und ruft hier wieder das Programm *make* auf.

Die Diskettenerzeugung läuft im Normalfall relativ zügig ab. Manchmal bleibt der Make-Lauf aber mit der Meldung stehen, daß ein Kernel keine Symboltabelle enthält. Dann braucht man nur den Lauf wiederholen. Im Regelfall funktioniert es beim zweiten (und allerspätestens im dritten) Lauf dann doch.

Mit einigen Kernel-Dateien und den Floppies allein kann man aber noch keine Distribution bauen. Es fehlen die Sets. Hier muß man jetzt entscheiden, ob man die äußerst zeitaufwendige Prozedur des kompletten Neubaus des Betriebssystems vornimmt oder die kurze Variante, indem man das installierte System übernimmt. Das wird über eine Environment-Variable gesteuert. Außerdem muß jetzt festgelegt werden, wohin die Daten kopiert werden sollen. Die Umgebungsvariable *RELEASEDIR* ist für das Ziel zuständig, *DESTDIR* für die Quelle. Nehmen wir als *DESTDIR* erst einmal »/usr/src«, also das Quelltextverzeichnis, als *RELEASEDIR* das Verzeichnis »/pub/release«:

```
# EXPORT DESTDIR=/usr/src
# EXPORT RELEASEDIR=/pub/release
```

Alternativ könnten die beiden Einträge genauso in der Datei »/etc/mk.conf« erfolgen. Diese Datei wird bei allen Make-Läufen eingelesen und ausgewertet. Hier der Inhalt einer solchen Datei:

```
# Set-Verzeichnisbaum:
RELEASEDIR=/pub/release/i386
# Binärdateien des neuen Systems
DESTDIR=/pub/dest/i386
# Verzeichnis für Temporärdateien während des Build:
BSDOBJDIR=/pub/release/obj
# Gesonderte OBJ-Verzeichnisse anlegen, nicht mischen:
MKOBJ=Yes
# Plattformbezogenes Build-Verzeichnis anlegen:
OBJMACHINE=YES
PKG_SUFX=.tbz
ACCEPTABLE_LICENSES+=fee-based-commercial-use
ACCEPTABLE_LICENSES+=adobe-acrobat-license
ACCEPTABLE_LICENSES+=no-commercial-use
ACCEPTABLE_LICENSES+=no-profit
```

Der obere Block dient dem Erzeugen der Sets, der untere Block ist für die Erzeugung von Drittanbieter-Paketen zuständig und hier nicht von Bedeutung.

Insgesamt stehen die in der folgenden Liste beschriebenen Umgebungsvariablen für *make* zur Verfügung. Während der erste Teil auch bei einem Teil-Build verarbeitet wird, ist die Steuerung mit den weiteren Variablen nur bei einem Komplett-Build möglich:

BUILDID	Name für den Build. Der hier angegebene Name wird an die Namen der OBJ-Verzeichnisse angehängt. Es lassen sich so spezifische Builds mit unterschiedlichen Build-Flags voneinander trennen. Die Variable ist normalerweise nicht gesetzt.
DESTDIR	Das Verzeichnis, das nach dem Kompilieren der Programme und Bibliotheken den Verzeichnisbaum des erzeugten Systems aufnimmt. Gibt man hier nichts an, wird direkt das installierte NetBSD-System überschrieben. Das ist dann praktisch, wenn das System direkt aus den Quellen aktualisiert werden soll. Bei einem Release-Build sollte hier ein sinnvoller Pfad auf eine Partition mit ausreichendem Platz weisen. Die Zielpartition muß Links und lange Dateinamen erlauben. Wenn die Umgebungsvariable *USETOOLS* auf *yes* gesetzt ist, ist *DESTDIR* auf einen leeren String (»überschreibe aktuelles System«) gesetzt, sonst ungesetzt. Bei einem vollen Build muß diese Variable gesetzt sein, sonst wird der Make-Lauf nicht durchgeführt.
MAKECONF	Der Name der generischen Make-Konfigurationsdatei. Diese Variable kann nur in der Umgebung definiert werden, da damit ein alternativer Name für die »/etc/mk.conf'« angegeben wird.
MKCATPAGES	Diese normalerweise auf *yes* gesetzte Variable legt fest, ob die Manpages erzeugt werden.
MKCRYPTO	Mit dieser normalerweise (mit *yes*) eingeschalteten Variable wird festgelegt, ob die Verschlüsselungsroutinen eingebunden werden sollen. Sie hat, wenn ausgeschaltet, keinen Einfluß auf die Paßwort-Verschlüsselung.
MKDOC	Diese normalerweise mit *yes* eingeschaltete Variable steuert, ob die Dokumentation für »/usr/share/doc« beim Build ebenfalls installiert werden soll.
MKHOSTOBJ	Diese Variable ist nur beim Cross-Compile von Bedeutung. Sie ist normalerweise mit *no* ausgeschaltet und wird mit *yes* eingeschaltet. *MKHOSTOBJ* legt fest, ob der Name, die Version und Architektur des Hostsystems an die OBJ-Verzeichnisse angehängt wird. Das ist nur sinnvoll, wenn auf einer Maschine für mehrere Plattformen übersetzt werden soll.
MKINFO	Diese Variable ist im Normalfall auf *yes* gesetzt. Sie legt fest, ob die GNU-Info-Dateien erzeugt und installiert werden sollen.

MKLINT	Diese Variable ist normalerweise auf *yes* gesetzt. Sie legt fest, ob *lint* beim Build gegen Teile des Quelltexts ausgeführt werden soll und ob die *lint*-Bibliotheken in »/usr/libdata/lint« angelegt werden sollen.
MKMAN	Hiermit wird festgelegt, ob die Man-Pages beim Build installiert werden sollen. Voreinstellung ist *yes*.
MKNLS	Hiermit wird angegeben, ob die Zonendateien kompiliert und installiert werden sollen. Voreinstellung ist *yes*.
MKOBJ	Diese wieder recht wichtige Variable legt fest, ob beim Lauf von *make obj* getrennte obj-Verzeichnisse angelegt werden sollen. Die Voreinstellung ist *yes* und sollte beibehalten werden, da die Interimsdateien sonst im normalen Sourcetree angelegt werden, was recht unübersichtlich ist.
MKPIC	Die Variable ist normalerweise (außer bei der Plattform *sh3*) auf *yes* gesetzt. Damit werden die Shared Objects und Bibliotheken dynamisch gelinkt. Setzt man die Variable auf *no*, wird das aufzubauende System statisch gelinkt.
MKPICINSTALL	Damit wird festgelegt, ob die Formatbibliotheken von *ar* (»lib*_pic.a«) für das Generieren von Shared Libraries beim Build installiert werden sollen. Die Voreinstellung ist *yes*.
MKPROFILE	Es wird festgelegt, ob Profile-Bibliotheken (»lib*_p.a«) erzeugt und installiert werden sollen. Diese Variable sollte normalerweise nicht verändert werden. Sie ist bei den meisten Plattformen auf *yes* gesetzt, wird aber bei manchen Ports abgeschaltet, da hierbei Probleme auftreten.
MKSHARE	Hiermit wird angegeben, ob die Dateien in »/usr/share« erzeugt und installiert werden sollen. Wird diese normalerweise auf *yes* gesetzte Variable auf *no* umdefiniert, werden damit automatisch auch die Variablen *MKCATPAGES*, *MKDOC*, *MKINFO*, *MKMAN* und *MKNLS* auf *no* geschaltet.
TOOLDIR	Das Verzeichnis für die Host-Tools. Der Name sollte für das gegebene Host-System eindeutig sein, allerdings dürfen verschiedene Zielsysteme das selbe *TOOLDIR* besitzen, da die zielabhängigen Programme eindeutige Namen haben. Die Variable ist normalerweise nicht gesetzt, dann werden die Dateien basierend auf den vom Programm *uname* zurückgegebenen Verzeichnis ».OBJDIR« in »src/tools« erzeugt.

UNPRIVED	Wenn diese Variable gesetzt ist, wird eine nicht-privilegierte Installation durchgeführt. Es werden dabei User- und Gruppenname, Zugriffsberechtigungen und Dateiflags nicht auf die installierten Dateien angewandt. Die werden vielmehr in die Datei *METALOG* in *DESTDIR* geschrieben. Der Inhalt dieser Datei wird anschließend für die Erzeugung der Distributions-Tar-Datei verwendet. Die Variable ist normalerweise nicht gesetzt.
UPDATE	Ist diese Variable gesetzt, wird bei allen Installationsarbeiten anhand des Zeitstempels der Dateien geprüft, ob eine Aktualisierung notwendig ist. *UPDATE* hat auch Auswirkungen auf Komplett-Builds und sollte nur mit Bedacht eingesetzt werden. Die Variable ist normalerweise nicht definiert.
USETOOLS	Gibt an, ob die Programme in *TOOLDIR* als Teil des Build-Prozesses verwendet werden sollen. Diese Variable muß bei Cross-Builds gesetzt sein. Sie kann die Werte *yes* (verwende die Tools aus *TOOLDIR*), *no* (verwende die Tools nicht, aber baue keine Komponenten, die diese Programme benötigen) und *never* (verwende die Tools nicht, auch wenn native Tool-Komponenten erzeugt werden). Normalerweise sollte die Variable nicht manuell geändert werden, da sonst Schwierigkeiten beim Build eines kompletten Verzeichnisbaums auftreten. Sie wird vom System je nach Art des Builds automatisch gesetzt.

Die folgenden make-Variablen haben nur Auswirkungen auf einen vollen Build des Sourcetrees. Sie haben keine Auswirkungen, wenn nur Teile des Verzeichnisbaums mit lokalen Make-Dateien erzeugt werden. Ihre Abfrage geschieht also nur in der Datei »/usr/src/Makefile«:

MKOBJDIRS	Es wird definiert, ob für die Interimsdateien automatisch gesonderte OBJ-Verzeichnisse beim *make obj*-Lauf angelegt werden. Die Voreinstellung ist *yes* und sollte beibehalten werden.
NOCLEANDIR	Ist diese Variable gesetzt, wird die *make cleandir*-Phase des Builds nicht durchgeführt. Interimsdateien eines früheren Laufs bleiben erhalten. Das beschleunigt einen Build, bei dem nur wenige Dateien geändert wurden, drastisch, führt aber gern zu Problemen. Die Variable ist normalerweise nicht gesetzt.
NODISTRIBDIRS	Verhindert, wenn gesetzt, den Lauf von *make distrib-dirs* des Builds. Das verhindert gleichzeitig den Aufruf von *mtree* auf das in *DESTDIR* definierte Zielverzeichnis. Die Definition ist nur auf Systemen sinnvoll, auf denen man keine *root*-Rechte besitzt und das Release als normaler User generiert.

NOINCLUDES	Wird diese Variable gesetzt, verhindert das den Lauf von *make includes* beim Build. Diese Option sollte beim Update des aktuellen Systems nicht gesetzt werden. Es ist besser, die Variable *UPDATE* zu definieren.
RELEASEDIR	Bei dieser Variable muß das Verzeichnis angegeben werden, in das die NetBSD-Sets nach dem Build-Prozeß im Lauf von *make release* kopiert werden. Diese Variable ist normalerweise nicht gesetzt, sie muß aber entweder in der Umgebung, in der »/etc/mk.conf« oder auf der Kommandozeile bei der Verwendung von *build.sh* angegeben sein, sonst bricht der Make-Lauf sofort ab.
UPDATE	Die Variable *UPDATE* hat beim Komplett-Build die selbe Bedeutung wie *NOCLEANDIR*. Die Erfahrung hat gezeigt, daß Build-Läufe mit vorhandenen Interimsdateien gern abbrechen. Das Setzen von *UPDATE* auf den Wert *yes* sollte also möglichst nicht vorgenommen werden.

Erfolgt das Bauen der Release über den kompletten Baum werden die Distributionssets automatisch erzeugt. Werden nur Teile des Baums verarbeitet, ist dem nicht so. Deshalb müssen in so einem Fall die Zielverzeichnisse angelegt und die Distributionstools komplett manuell erzeugt werden:

```
# mkdir pub/release/i386
# mkdir pub/release/i386/installation
# mkdir pub/release/i386/binary
# mkdir pub/release/i386/binary/sets

# cd /usr/src/distrib/notes
# make
# cd /usr/src/distrib
# make
# cd /usr/src/distrib/sets
```

Nachdem die ganzen Dateien erzeugt wurden, werden jetzt aus dem laufenden System die Sets erzeugt. Das ist nicht besonders elegant, funktioniert aber:

```
# export DESTDIR=/
# make sets
```

Hier ist zu beachten, daß es nicht möglich ist, die X-Sets direkt aus dem X-Sourcen zu erzeugen, deshalb müssen sie immer aus den installieren Binärdateien des Systems zusammengesammelt werden. Leider geht das (wenigstens bei NetBSD bis Version 1.6) nicht mit dem *make*-Befehl. Es müssen die Low-Level-Befehle aufgerufen werden:

```
# sh maketars xserver xbase xcomp xfont
# mv pub/release/i386/*tgz /pub/release/i386/binary/sets
# make makesums
```

Diese hier manuell aufgerufenen Befehle werden sonst automatisch von der Make-Datei gestartet. Das Verschieben der Dateien vor dem *make makesum* ist nötig, weil die X-Sets vom Skript eine Verzeichnisebene zu tief angelegt wurden.

Damit sind die Arbeiten nahezu beendet, es müssen noch die Floppy-Images und die Kernel-Dateien umkopiert werden. Das war's dann.

Diese Arbeiten wurden auf dem i386-Port vorgenommen. Auf anderen Plattformen funktioniert es ähnlich. Allerdings kann so kein Cross-Compile durchgeführt werden. Früher oder später – selbst wenn man den Cross-Compiler aus den Package-Sourcen installiert hat – bricht der Lauf ab. Ab NetBSD 1.6 wurde die Vorgehensweise deshalb drastisch vereinfacht.

16.2 build.sh und Cross Compile

Ab NetBSD 1.6 enthält das Verzeichnis »/usr/src« eine Datei »build.sh«. Mit diesem Skript ist es problemlos möglich, die Grund-Sets für fast beliebige Zielplattformen auf einem anderen NetBSD-System zu erzeugen. Der Sinn ist klar: Wer ein NetBSD auf unterschiedlich schnellen Plattformen hat, wird nur einen Port installieren (und zwar den schnellsten), die Quellen herunterladen und sich die anderen Ports selbst übersetzen. Das funktioniert für das Basis-System ganz hervorragend. Testweise wurden auf einer i386-Maschine die Ports für Shark, Mac68k und MacPPC erzeugt. Das Kompilieren dauerte auf einem AMD K6/2 400 jeweils etwa sechs bis sieben Stunden. Gerade auf dem Macintosh mit 68040/25-CPU wäre es ein vielfaches der Zeit gewesen. Auf einem moderneren Rechner als dem verwendeten ginge es natürlich auch erheblich flotter.

Für die Arbeiten sind die folgenden drei Einträge in der Datei »/etc/mk.conf« unabdingbar:

```
BSDOBJDIR=/pub/release/obj
MKOBJ=YES
OBJMACHINE=YES
```

Hiermit wird gesteuert, daß kein gemeinsames Verzeichnis für die obj-Dateien erzeugt wird (*OBJMACHINE*), die obj-Verzeichnisse angelegt werden (*MKOBJ*) und wo die Arbeitsprogramme abzulegen sind. Dieses Verzeichnis muß angelegt sein, sonst steigt der Build-Lauf gleich zu Beginn aus. Was auf keinen Fall gesetzt sein darf, ist der Schalter

```
UPDATE=YES
```

Die Erfahrung hat gezeigt, daß bei der Angabe dieses Schalters unvollständige Läufe generiert werden. Es ist ausgesprochen ärgerlich, wenn ein Make-Lauf aber nach fünf bis sechs Stunden abbricht.

Nun wechselt man in das Verzeichnis »/usr/src« und ruft folgenden Befehl auf:

```
# ./build.sh -m <Plattform> -d -D /pub/release/<plattform> \
#            -R /pub/release/<plattform>
```

Die Parameter *-D* und *-R* entsprechen dabei Eins-zu-Eins den Umgebungsvariablen *DESTDIR* und *RELEASEDIR*. Das Build-Skript kennt die folgenden Kommandozeilenschalter:

-a arch	Setzt die Variable *MACHINE_ARCH* auf *arch*.
-B buildid	Setzt die Variable *BUILDID* auf *buildid*. Damit wird auch die Kennung des Builds an den Namen des Wrapper-Make-Skripts angehängt. Der daraus entstehende Name ist dann in der Form »nbmake-MACHINE-BUILDID«.
-b	Erzeuge »make« und schreibe ein Skript »nbmake-MACHINE«.
-d	Baue eine Komplettdistribution. Das unterscheidet sich von einem normalen Build dahingehend, daß auch die Dateien in »etc« installiert werden. Das bedeutet aber nicht, daß mit diesem Schalter das Erzeugen eines Release verbunden wäre. Es werden keine Sets in *RELEASEDIR* generiert.
-j njob	Dieser Parameter wird an *make* weitergereicht und dient der Steuerung des Build-Ablaufs. Er ist normalerweise nicht erforderlich.
-m mach	Setzen des Werts der Variable *MACHINE* auf den Wert *mach*. Diese Angabe überschreibt den Wert von *MACHINE_ARCH* in der Prozeß-Umgebung, wurde der Parameter *-a* nicht angegeben. Alle Cross-Build benötigen den Parameter *-m*. Ist er nicht gesetzt, wird *MACHINE* des Hosts automatisch ermittelt und gesetzt. Bei einem normalen Build ist er deshalb nicht nötig.
-n	Mit diesem Parameter werden nur die von *build.sh* beim entsprechenden Lauf auszuführenden Befehle angezeigt, aber kein echter Lauf durchgeführt. Dieser Aufruf entspricht also einem Aufruf von *make -n*.
-o	Entspricht dem Setzen der Umgebungsvariable *MKOBJDIRS* auf *no*.
-r	Entfernen des Inhalts von *DESTDIR* und *TOOLDIR* vor dem Build-Lauf. Damit wird ein sauberer Start gewährleistet. Das Löschen von *DESTDIR* wird beim Build für das native System auf das Root-Verzeichnis aber verhindert.

-t	Nur der Build und die Installation der Host-Tools aus »src/tools« wird durchgeführt. Diese Option verlangt die zusätzliche Angaben von *-b*.
-u	Setzt die Variable *UPDATE*.
-w wrapper	Es wird dafür gesorgt, daß das *nbwrapper*-Skript an einer definierten Position (normalerweise im Suchpfad) abgelegt wird. Hier muß dann der komplette Namen (Verzeichnispfad und Dateiname) angegeben sein. Normalerweise ist das Setzen dieses Schalters nicht notwendig.
-D dest	Setzt die Variable *DESTDIR* auf *dest*.
-M obj	Setzt die Variable *MAKEOBJDIRPREFIX* auf *obj*.
-O obj	Erzeugt ein Makro für *MAKEOBJDIR*, mit dem die Position der Build-Zwischendateien definiert wird. Wenn der Wert auf »/usr/obj« gesetzt ist, werden die Build-Dateien in die Verzeichnisse »/usr/obj/bin«, /usr/obj/lib und so weiter abgelegt.
-R rel	Setzen der Variable *RELEASEDIR* auf den Wert *rel*. Ist diese Option gesetzt, wird kein *make build*, sondern ein *make release* durchgeführt. Bei Cross-Kompiliervorgängen ist diese Variable obligatorisch.
-T tools	Setzt die Variable *TOOLDIR* auf *tools*. Ist diese Variable gesetzt, führt *make* einen Rebuild nur durch, wenn das benötigt wird.
-U	Setzt die Variable *UNPRIVED*.

Mit den Kenntnissen über die möglichen Parameter ist es einfach, die entsprechenden Daten anzugeben. Für *Plattform* ist jedesmal die Zielplattform einzusetzen, beispielsweise i386, mac68k, macppc, sun3, cobalt oder eine der anderen, die in der Datei »build.sh« in der Funktion *getarch()* aufgeführt ist.

Wichtig ist, daß der Sourcetree beim Start sauber ist, das heißt, keine eigenen Links gelegt und Dateien umbenannt wurden. Außerdem sollten keine Verzeichnisleichen früherer Fehlversuche vorhanden sein. Das führt – insbesondere wenn man den Cross-compiler-Lauf zum Erzeugen der eigenen Plattform verwendet (hier macht man am ehesten Änderungen) – zum Abbruch.

Der Lauf von »build.sh« ruft *make* mit den richtigen Parametern auf und durchläuft die Verzeichnisse in der korrekten Reihenfolge. Aber – es werden nur die normalen Sets und nicht die X-Sets erzeugt. Um für die eigene Plattform die X-Sets zu erzeugen, muß wieder der Lauf von

```
# cd /usr/src/distrib/sets
# sh maketars xserver xbase xcomp xfont
# mv pub/release/i386/*tgz /pub/release/i386/binary/sets
# make makesums
```

erfolgen. Das ist nicht besonders schön. Man kann zwar versuchen, die X-Quelltexte einzuspielen und zu kompilieren. Das funktioniert auch für die eigene Plattform. Aber ein *make World* von X erzeugt und installiert die Programme direkt im Dateisystem, was ja eigentlich in diesem Fall nicht erwünscht ist. Außerdem ist weder ein Cross-Compile der X-Dateien noch der Ports möglich. Ersteres sollte zwar theoretisch funktionieren, denn X ist in der Tat für Cross-Compile ausgelegt. Es scheitert aber an brauchbaren Dokumentationen. Das zweite ist generell nicht möglich, da, selbst wenn das Übersetzen gelingen mag, NetBSD die erzeugten Binärdateien zuerst installieren und dann aus den installierten Dateien packetieren will. Allein diese Vorgehensweise schließt das Übersetzen für eine andere Plattform aus.

A Spezielle Anwendungsbereiche

A.1 Schreibgeschützter Rootbereich

von Ignatios Souvatzis

Der Rootbereich – die Wurzel des Dateibaums – enthält traditionell das Minimum an Programmen und Datenbereichen, um ein unix-artiges System so weit zu starten, daß weitere Plattenbereiche montiert werden können. Während Programme nicht schreibbar zu sein brauchen, gilt dies allerdings für die Datenbereiche – offensichtlich ist das bei Protokolldateien, Spooldateien für Mail, Drucker, Fax. Dazu kommen noch eine Reihe subtilerer Schreibzugriffe, auf die weiter unten eingegangen wird.

In einer klassischen Installation bereitet der Schreibzugriff auf den Rootbereich keine Probleme. Aber es gibt Anwendungen, bei denen ein schreibgeschützter Rootbereich Sinn machen würde:

- Ein Demonstrationssystem auf CD-ROM. Hier kann nicht geschrieben werden.
- Ein Firewallrechner oder ein sonstiges sehr exponiertes System. Wenn Konfigurationsdateien nicht schreibbar sind, kann auch ein gehacktes System keinen dauerhaften Schaden anrichten. Wenn Sie den Verdacht haben, daß ein Einbruch stattfindet, brauchen Sie nur den Rechner neu zu starten. Wenn keine Dateien schreibbar waren, sind die Filesysteme sauber, so daß kein *sync* beim Shutdown und kein Filesystemcheck beim Reboot stattfinden – die Rebootzeit ist gering und alle etwaigen Manipulationen sind fürs erste wieder getilgt. Die Programmdateien selbst waren natürlich gar nicht erst veränderbar. Eine weitere Sicherheitssteigerung ergibt sich, wenn alle weiter unten beschriebenen schreibbaren Dateisysteme als *noexec* angelegt werden – dann kann ein Einbrecher auch kein neues Programm kompilieren, installieren oder starten. Natürlich muß der Einbruchsweg so schnell wie möglich verschlossen werden, aber die komplette Neuinstallation können Sie sich für's erste sparen.
- Ein gemeinsames Filesystem, aus dem mehrere im wesentlichen identische Netzwerk-Computer gebootet werden. Es führt es zu diversen Verwirrungen, wenn die Rechner sich gegenseitig ihre Dateien manipulieren.

Protokolldateien in »/var/log« und Spooldateien für E-Mail, Drucker und so weiter wurden ja schon erwähnt. Weitere Problemkinder sollen hier aufgezählt werden.

- Daemonen neigen dazu, in »/var/run« die »*.pid«- Dateien mit ihrer Prozeßnummer und teilweise weiteren Informationen abzulegen.
- Manche Daemonen legen hier (in historischen BSD-Systemen und einigen anderen sogar in »/etc« oder »/dev«) Socket- oder Pipe-Pseudodateien an, und zwar jedesmal, wenn sie starten. Darüber kommunizieren ihre Clients mit ihnen. Ein Beispiel ist *lpd*; *lpr* benachrichtigt *lpd* über ein an traditionell »/dev/printer«, bei NetBSD »/var/run/printer« geschicktes Kommandopaket, daß ein neuer Druckauftrag vorliegt, und *lpc* steuert *lpd* bei Bedarf auf dem gleichen Weg. Die syslog-Library-Funktionen (*open-*

log() und Verwandte) benutzen ebenfalls einen Socket (traditionell »/dev/log«, bei NetBSD »/var/run/log«), um die Meldungen an den *syslogd* zu schicken.

- Die Geräte-Pseudodateien in »/dev« müssen ebenfalls oft schreibbar sein. (Wohlgemerkt – nicht der Zugriff auf das Gerät, sondern der Eintrag in »/dev« selbst!) Terminaldateien ändern jedesmal ihren Zeitstempel, wenn beispielsweise ein neuer Lesevorgang gestartet wird, die Inaktivitätserkennung von Bildschirmschonern und *who*, *finger* und ähliches funktioniert auf diese Weise. Zusätzlich müssen *login*, *getty* und *sshd* das Terminal oder Netzwerkterminal des Benutzers für die Zeit ihrer Sitzung »schenken« (besser: verleihen), und es ihr hinterher wieder wegnehmen. Bei gemeinsam genutzten Rootsystemen führt das zur Verwirrung, bei schreibgeschützten funktioniert das nicht.
- Der *shutdown*-Prozeß legt »/etc/nologin« an, der Rebootprozeß löscht es wieder.
- Der Administrator möchte gelegentlich administrative Dateien in »/etc« löschen.
- *newsyslog* rotiert regelmäßig die Protokolldateien.

Ein schreibgeschütztes Rootsystem anlegen

Zunächst einmal sollte alles abgeschaltet werden, was Sie nicht benötigen, aber Schreibzugriffe durchführen würde. Zum Beispiel können Sie »/etc/syslog.conf« so umschreiben, daß alle Protokolle auf der Konsole und/oder einem entfernten Rechner landen und darauf verzichten, die Bootzeit und die Kernelversion nach »/etc/motd« schreiben zu lassen.

Dies funktioniert jedoch nicht, wenn Sie beispielsweise »/var/run«, ein Druck- oder Mailspool und so weiter benötigen. Diese können Sie sich jedoch als Memory File System anlegen, Beispiele siehe unten. Das MFS von 4.4BSD (und damit NetBSD) benutzt als Speichermedium den virtuellen Adreßraum des Programms *mount_mfs*, das gleichzeitig das Programm *newfs_mfs* ist, ist also auslagerungsfähig. Sie benutzen also nur soviel Hauptspeicher, wie tatsächlich grade nötig ist, jedenfalls dann, wenn Sie einen Swapbereich auf Festplatte oder NFS konfiguriert haben und der Hauptspeicher zum Engpaß wird.

Beachten sollten Sie, daß Druck- und Maildateien im MFS bei einem Systemabsturz verloren gehen. (Manche Systemadministratoren halten dies für einen Vorteil...) Falls das nicht akzeptabel ist, können sie diese Bereiche von einer lokalen Festplatte oder über das Netz montieren. Bei einem Demosystem auf CD-ROM wäre das aber durchaus akzeptabel, ein reiner Router/Firewall kann auf diese Funktionen ganz verzichten.

Ein besonderes Problem ist das Verzeichnis »/dev«. Es muß schreibbar sein, soll aber beim Systemboot mit einem vordefinierten Sortiment von Gerätedateien belegt sein. Hier kann man ein MFS mounten und danach eine Kopie der */dev/MAKEDEV*-Skripte ausführen. Bis NetBSD 1.5.x war dazu eine spezielle Startup-Datei notwendig, seit NetBSD 1.6 übernimmt das dankenswerterweise das Programm */sbin/init* selbst, das vom Kernel als erstes gestartet wird, sofern es »/dev/console« nicht findet.

Unter Umständen – je nach Bedarf – ist es am einfachsten, auf »/« ein hinreichend großes MFS »durchscheinend« zu mounten:

```
mount -t mfs -o union [-o weitere optionen ...] swap /
```

Der Autor hat das bei seiner Anwendung jedoch nur für »/etc« getan, um ein schreibbares »/etc/nologin« und »/etc/motd« zu erhalten.

Fallstudie: Schreibgeschütztes Rootfilesystem für eine Gruppe plattenloser Netzwerkrechner

Der Autor hatte vor einigen Jahren die Aufgabe, zehn plattenlose Unix-Rechner vom Typ *DNARD* in ein bestehendes Rechnernetz zu integrieren und für ein Parallelrechnerpraktikum an einer Universität zur Verfügung zu stellen. Die Benutzer sind halbjährlich wechselnde Studierende, die nicht näher bekannt oder besonders vertrauenswürdig sind. Zu allem Überfluß sollten sie sich auch vom allgemeinen Studentenpool der Informatik oder von zu Hause aus auf diese Systeme einloggen können.

Die Rechner sind alle identisch und bekamen ein gemeinsames Rootfilesystem. Bei nur 64 Mbyte RAM wurde je ein Swapbereich über das Netz vom Fileserver montiert, ebenso ist je Rechner ein »/var«-Bereich auf dem Fileserver vorhanden.

Die Rechner bekommen ihre Identität über das *Dynamic Host Configuration Protocol* (DHCP) mitgeteilt, das auch schon für das Booten vom Netzwerk benötigt wird. Es wurde darauf verzichtet, eine je Rechner getrennte SSH-Identität (ssh host key) bereitzustellen – die Rechner brauchen nicht kryptographisch sicher unterschieden zu werden. Benutzeridentitäten werden über die Network Information Services (NIS) geladen, das Filesystem mit den Benutzerdateien über NFS.

Im folgenden sollen einige zentrale Dateien aus diesem Setup vorgestellt werden:

Die »/etc/fstab«:

```
server:/BSD/root2/1.5     /           nfs ro                0 0
server:/BSD/root/vargames /var/games  nfs rw                0 0
swap                      /tmp        mfs rw,-s=32768       0 0
swap                      /var/run    mfs rw,-i=512,-s=256 0 0
```

Hier fehlen die Homebereiche, die im konkreten Fall vom Automounter verwaltet werden.

Die »/etc/rc.conf« sei hier vollständig wiedergegeben:

```
rc_configured=YES
shroot=YES
shroot_pfx_var=server:/BSD/root/var-
shroot_pfx_swap=server:/BSD/swap/
critical_filesystems_beforenet="/var /var/run"
update_motd=NO
rpcbind=YES # nfs
domainname=nis.doma.in # NIS
```

```
ypbind=YES
amd=YES amd_dir=/var/amdroot
nfs_client=YES
ip6mode=autohost
defaultroute=10.10.10.10
sshd=YES postfix=YES ntpd=YES
lpd=YES lpd_flags=-s
inetd=YES       # ntalk, (c)fingerd
```

Das wichtigste ist *critical-filesystems-beforenet*, das dafür sorgt, daß »/var« und »/var/run« tatsächlich frühzeitig eingebunden werden, und natürlich die Freigabe von NFS über die Zeilen *nfs_client* und *rpcbind*.

Weiterhin bedeutsam sind die beiden *sbroot*-Präfixe. Das Skript */etc/rc.d/sbroot* (Listing Listing A.1) baut sich daraus, und aus dem Rechnernamen die Pfade für den Swapbereich und »/var«.

Beachten Sie, wie dieses Skript den Rechnernamen erfährt. Dies geschieht durch den Aufruf von *hostname* aus dem Kernel, der ihn wiederum über DHCP erfahren hat, als er sich über DHCP nach der Rechner-IP-Adresse und dem Rootfilesystem erkundigt hat.

REQUIRE: root ordnet dieses Skript im Bootvorgang nach dem Mouten von »/« ein, *BEFORE: mountcritlocal* dagegen sorgt unter anderem dafür, daß die oben erwähnten *critical-filesystems-beforenet* erst nach Ablauf dieses Skripts gestartet werden; insbesondere ist also der Swap schon konfiguriert, was wichtig für die diversen MFS-Mounts ist.

Wie aber benutzt man bei dieser Konfiguration das pkg-System zur Installation von Drittsoftware? Es sollte ausreichen, einem Client zeitweilig Schreibzugriff auf »/usr/pkg« zu geben, die Environment-Variable *PKG_DBDIR* auf »/usr/pkg/pkgdb« zu setzen und alles Nötige zu installieren. Wenn *PKG_DBDIR* so gesetzt bleibt, können die Benutzer hinterher von jedem Rechner aus mit *pkg_info* die Liste der installierten Packages erhalten.

Weitere Dokumentation zum Thema:

Manualpages rc(8), rc.conf(8), rcorder(8) und entsprechende Kapitel dieses Buchs zum Startupsystem von NetBSD

Manualpages mount_mfs(8) und newfs_mfs(8) zum Memory Filesystem

```
#!/bin/sh

# PROVIDE: shroot
# REQUIRE: root
# BEFORE: mountcritlocal

# shared root setup.

. /etc/rc.subr

name="shroot"
start_cmd="shroot_start"
stop_cmd=":"
required_files="/sbin/MAKEDEV /sbin/MAKEDEV.local"

shroot_start () {

    hostname=`/bin/hostname`

    case "$shroot_pfx_var" in
        "") ;;
        *) /sbin/mount -t nfs ${shroot_pfx_var}${hostname} /var ;;
    esac
    case "$shroot_pfx_swap" in
        "") ;;
        *) /sbin/mount -t nfs ${shroot_pfx_swap}${hostname} /swap\
            && /sbin/swapon /swap ;;
    esac

    /sbin/mount -t mfs -o -i=256 -o -s=512 swap /dev
    /sbin/mount -t mfs -o -i=256 -o -s=512 -o union swap /etc
    /bin/chmod 755 /dev /etc
}

load_rc_config $name
run_rc_command "$1"
```

Listing A.1: Die Datei /etc/rc.d/shroot

A.2 Die CD-ROM zum Buch: NetBSD Live!

Diesem Buch liegt eine Live-CD-ROM bei, die entgegen den sonstigen Gepflogenheiten, daß man einem Buch zu einem freien Betriebssystem eine Version zur Installation beilegt, die dann schneller veraltet ist, als das Buch gedruckt ist, eine Demonstration der Fähigkeiten von NetBSD darstellt, aber auch als komfortables Rettungs- und Reparatursystem eingesetzt werden kann.

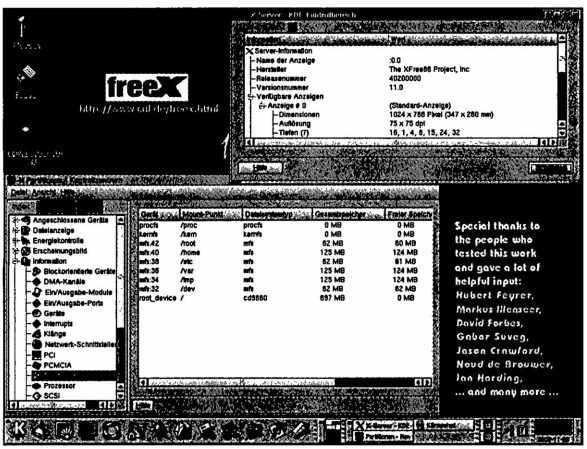


Abb. A.1: Die Oberfläche von NetBSD Live! mit deutschen Locales

Das System besitzt den Namen NetBSD Live!, liegt in der Version 0.5 vor und funktioniert auf einer großen Zahl von Hardwarezusammenstellungen sehr gut. Es ist möglich, X zu starten und KDE 3.0.3 inklusive der Anwendungen KOffice, Gimp und Mozilla zu benutzen. Das System ist netzwerkfähig, wenn ein DHCP-Server zur Verfügung steht. Den Umgang mit dem System zu beschreiben, ist nicht Sinn dieses Anhangs, die CD-ROM enthält eine Dokumentation im PDF- und HTML-Format im Verzeichnis »/doc«, die bei Interesse studiert werden kann. Hier soll es darum gehen, was ein solches System prinzipiell von einem unterscheidet, das von Festplatte bootet, und was beim Bau einer solchen Distribution zu beachten ist. Einige dieser Themenbereiche wurden in diesem Buch in anderem Zusammenhang bereits angesprochen, andere sind ausschließlich für ein CD-Medium von Belang.

A.2.1 Der Kernel und das Booten

Der generische NetBSD-Kernel umfaßt fast alles an Geräten, was auf durchschnittlicher Hardware benötigt wird. Ein auf nur einer Maschine laufender Betriebssystem-Kern wird im Regelfall dann an diese Maschine angepaßt. Das bringt Speicherersparnis und bedeutet einen schnelleren Bootvorgang. Ein Kernel, der von einem austauschbaren Medium bootet, benötigt aber mehr als die Standard-Geräte einer bestimmten Hardware-Konfiguration, in ihm sollten möglichst viele Geräte enthalten sein. Zudem ist es

entscheidend, daß der Kernel von einem Medium bootet, das nicht im BSD-typischen FFS-Dateisystem formatiert ist. CD-ROMs besitzen das ISO-9660-Dateisystem, unter Unix mit den sogenannten RockRidge-Erweiterungen, die für lange Dateinamen zuständig sind.

Es gibt grundsätzlich zwei Möglichkeiten, von so einer CD zu booten. Zum einen, indem man von einer FFS-Bootdiskette den Kernel startet, der dann auf die CD wechselt. So arbeitet auch das NetBSD-Installationssystem. Oder man bootet ein anderes Betriebssystem, das dann einen NetBSD-Loader startet.

Die zweite Lösung ist einfacher zu implementieren, kann aber aus lizenzrechtlichen Gründen nicht distribuiert werden. Sie wird hier zuerst beschrieben.

Man benötigt ein DOS-System, besonders eignet sich Caldera OpenDOS 7.01, da es für nicht-kommerzielle Zwecke verwendet werden darf. Man lädt sich das Archiv des Betriebssystems von *ftp://ftp.lineo.com/pub/drdos/ OpenDOS.701* und erzeugt eine Boot-Diskette, die außer dem Betriebssystem-Kernel den CD-Treiber »nwcdex.exe« und einen ATAPI-CD-ROM-Treiber enthält. Außerdem kann man – sofern man solch einen Treiber zur Hand hat und benötigt – noch einen SCSI-Gerätetreiber auf die Diskette kopieren. Einen guten ATAPI-CD-ROM-Treiber für MS-DOS findet man in Form des Mitsumi-Treibers, der auch aus dem Internet von *http://download.com.com/3000-2100-901550.html?legacy=cnet* geladen werden kann. Er funktioniert auch mit anderen DOS-Betriebssystemen. Informationen zu SCSI-Treibern erhält man im Internet auf »Bart's page about ASPI« bei *http://www.nu2.nu/aspi/*.

Die Diskette benötigt außerdem noch je eine Datei »config.sys« und »autoexec.bat«, um die CD-Treiber zu laden:

```
LASTDRIVE=I
REM The Mitsumi ATAPI driver. You can use any other one you have:
device=mtmcdai.sys /D:ATAPI
```

Listing A.2: Die für die Bootdiskette benötigte Datei config.sys

```
@echo off
REM AUTOEXEC.BAT
REM This is the DR DOS driver. Alternatively we can use Microsofts's
REM MSCDEX. It uses the same command line parameters. The parameter
REM /D: must be the same as the parameter of the driver in the the
REM config.sys file.
nwcdex /D:ATAPI /L:I
REM This booter is provided with NetBSD:
dosboot i:\netbsd
```

Listing A.3: Die für die Bootdiskette benötigte Datei autoexec.bat

In der »autoexec.bat« wird zum Schluß noch das Programm *dosboot.com* aufgerufen. Es ist ein NetBSD-Loader, der das gestartete DOS-Betriebssystem verdrängt und den auf der Kommandozeile angegebenen NetBSD-Kern bootet. Dieses Programm ist bei jeder NetBSD/i386-Distribution enthalten. Falls die CD-ROM mehrere verschiedene Kernel-Dateien enthält, läßt sich an anstelle des direkten Aufrufs auch ein Bootmenü aufbauen, in dem dann der zu startende Kernel abgefragt wird.

Wichtig ist in den beiden DOS-Startdateien, daß für das Einbinden des CD-Laufwerks ein recht hoher Laufwerksbuchstabe eingetragen wird, der auch dem CD-Treiber fest zugewiesen ist. Sollten sich auf der oder den eingebauten Festplatten bereits DOS-Partitionen befinden, werden diese sonst vor dem Minimal-Betriebssystem vor dem CD-Laufwerk eingebunden und der Laufwerksbuchstabe ist damit eventuell nicht zuordenbar.

Von dieser Diskette kann ein NetBSD von einer beliebigen CD-ROM, auf der sich ein Kernel mit dem Namen »netbsd« befindet, gestartet werden.

Das macht aber nur Sinn, wenn dieser Kernel auch dazu fähig ist, vom ISO9660-Dateisystem zu starten. Er benötigt dazu nur die folgenden Zeile in der Konfigurationsdatei

```
# Kernel root file system and dump configuration.
config          netbsd  root on cd0d type cd9660
```

anstelle des normalen Eintrags. Wenn Sie anstelle eines Rettungs- und Demo-Systems einen CD-basierten Router mit NetBSD als Betriebssystem bauen wollen, müssen Sie hier das entsprechende Security-Level einstellen. Ansonsten sind die durchzuführenden Arbeiten grundsätzlich die selben.

Nun kann dieser Kernel bereits gebaut und in ein mögliches Mastering-Verzeichnis kopiert werden. Tatsächlich sind hier nicht mehr Arbeiten erforderlich. Außerdem sollte man nicht immer von der Diskette starten müssen, also erzeugt man aus der fertigen Diskette ein Floppy-Image, das man ebenfalls ins Mastering-Verzeichnis kopiert.

Am einfachsten wird das Disketten-Image auf einem Unix-Rechner mit dem Befehl *dd* erzeugt:

```
# dd if=/dev/fd0a of=pub/master/boot.iso
```

Die Diskette muß aber genau 1,44 Mbyte oder 1,2 Mbyte groß sein. Das platzsparende Verwenden einer kleineren Diskette (360 Kbyte oder 720 Kbyte) geht nicht, da das El-Torito-Format für bootbare CDs diese kleinen Diskettenformate nicht unterstützt.

Erheblich aufwendiger ist es, wenn NetBSD nativ – das heißt nicht über den Umweg eines anderen Betriebssystems – gestartet werden soll. Die Kernel-Konfiguration sieht genauso aus, aufwendiger ist aber das Erzeugen der Bootdisketten.

Wie aufwendig man die Arbeit gestalten will, ist prinzipiell von der Geschwindigkeit des Rechners, auf dem die Arbeiten durchgeführt werden, abhängig. Auf einem Boliden

mit Gigahertz-Prozessor dauert es nicht sehr lange, eine komplette Release mit *build.sh* zu kompilieren. Dort werden alle im folgenden beschriebenen Schritte mehr oder minder automatisch durchgeführt. Auf langsameren Maschinen, bei denen das Release-Build mindestens mehrere Stunden dauert, sollte man sich auf das notwendigste beschränken.

Gebraucht werden unbedingt ein Sourcetree, der außer den Kernel-Sourcen auch die Distributionstools enthält, und ein neuer Kernel, der zwingend den Namen »INSTALL« trägt. Genutzt werden kann der oben schon beschriebene Generic-Kernel, der nur die geänderte Boot-Zeile enthält. Also muß die Konfigurationsdatei »/usr/src/sys/arch/i386/conf/INSTALL« gesichert werden und der neue CD-Kernel diesen Namen erhalten.

Geht man den bequemen Weg, wechselt man jetzt nach »/usr/src« und wirft *build.sh* an:

```
# cd /usr/src
# export DESTDIR=/pub/dest/i386
# export RELEASEDIR=/pub/release/i386
# ./build.sh
```

Nun heißt es warten.

Anschließend sollte man die beiden Umgebungsvariablen wieder löschen:

```
# unset DESTDIR
# unset RELEASEDIR
```

Als Boot-Image wird die Datei »/pub/release/i386/installation/floppy/boot-big.fs« verwendet (bei manchen Läufen heißt sie auch »boot-big1.fs«, das spielt weiter aber keine Rolle).

Es handelt sich hierbei um ein Diskettenimage mit einer Größe von 2,88 Mbyte, das ausschließlich als El-Torito-Bootimage der CD eingesetzt werden kann[1]. Wer sicher gehen will, kopiert auch noch die beiden Dateien »boot1.fs« und »boot2.fs« auf den Master. Wenn auf einem Rechner das BIOS standhaft sich weigert, vom CD-ROM-Laufwerk zu starten, lassen sich damit zwei Startdisketten generieren, mit denen das System immer noch gestartet werden kann.

Wessen Rechner zu langsam ist oder wer keine Lust hat, nur wegen eines Disketten-Images eines Komplettdistribution zu erzeugen, die man ohnehin anschließend wegwerfen kann, erzeugt die Diskette manuell.

---

[1] Es gab tatsächlich eine Zeitlang IBM-Maschinen mit 2,88 Mbyte-Diskettenlaufwerken. Das Format hat sich aber nicht durchgesetzt und heute kann man solche Laufwerke nicht mehr kaufen. Trotzdem ist dieses Format den PC-BIOSen ab Pentium-Boards bekannt.

Dazu müssen vorab alle Installations-Kerneldateien angelegt werden. Im Konfigurationsverzeichnis ruft man dazu

```
# for k in INSTALL* ; do config $k ; done
```

auf. Dann wechselt man nach »../compile« und erzeugt die neuen Kernels:

```
# for k in INSTALL* ; do cd $k; make depend && make ; cd .. ; done
```

Man den Rechner eine gute Zeit allein lassen. Es müssen immerhin außer dem benötigen auch alle anderen Installations-Kerneldateien (Laptop, Small, Tiny und so weiter) gebaut werden.

Wurden alle Kernel-Dateien erfolgreich erzeugt, wechselt man nach »/usr/src/distrib/i386/floppies« und ruft hier *make* auf. Aus den vorher erzeugten Kernel-Dateien werden in einer recht komplizierten Prozedur jetzt die diversen Diskettenimages erzeugt. Hier spielen die Umgebungsvariablen *RELEASEDIR* und *DESTDIR* keine Rolle. Die Images werden nicht umkopiert und befinden sich anschließend in den Unterverzeichnissen.

Sie sollten auch darauf achten, daß mögliche andere Umgebungsvariablen für den Release-Build nicht in der Datei »/etc/mk.conf« aktiv sind. Das führt nur zur Verwirrung von *make*.

Kopieren Sie die Floppy-Images in Ihr Mastering-Verzeichnis. Ein Single-User-Bootsystem ist hiermit schon fast fertig.

A.2.2 Das CD-Dateisystem

Wir haben bis jetzt einen bootfähigen Kernel mitsamt einer Bootdiskette. Was fehlt, ist das eigentliche CD-ROM-Dateisystem. Dessen Erzeugung ist auch nicht komplizierter.

Benötigt werden nur ein Verzeichnisbaum der Systemprogramme und ein angepaßtes »etc«-Verzeichnis. Dieser letzte Schritt ist mit Abstand der komplizierteste. Alles andere ist eigentlich einfach.

Fangen wir an, einen Verzeichnisbaum aufzubauen. Das Zielverzeichnis soll »/pub/ro-image« heißen. In diesem Verzeichnis werden die Unterverzeichisse »bin«, »etc«, »root«, »usr«, »sbin«, »dev«, »tmp«, »var« und »home« angelegt. Auf letzteres könnte man eigentlich verzichten, dann aber keinen User anlegen. In »usr« ist der übliche »usr«-Tree mit den Verzeichnissen »lib«, »libexec«, »share«, »bin« und »sbin« aufzubauen. Außerdem wird natürlich noch ein »usr/X11R6«, wieder mit den üblichen Unterverzeichnissen benötigt.

Das heißt, es wird ein kompletter Unix-Verzeichnisbaum aufgebaut. Es bestehen zwei Möglichkeiten, diesen Baum aufzubauen: entweder manuell oder durch rigoroses Einkopieren der Original-NetBSD-Installationssets, wobei die beiden Sets »comp« und »games« unbedingt weggelassen werden sollten. Sie kosten nur Platz. »games« besitzt

ohnehin mehr historischen Wert, kompilieren kann man auf einer CD nicht. »comp« benötigt aber enorm Platz.

Egal, für welche Variante Sie sich entscheiden, am Schluß muß ein komplettes NetBSD stehen. Sie können das ausprobieren, indem Sie mit *chroot /pub/ro-image* dorthin wechseln. Ein Warnung: Zum jetzigen Zeitpunkt funktioniert das nur, wenn Sie als Shell eine der Standard-Shells (*ksh*, *csh* oder *sh*) verwenden. Eine *bash* oder *zsh* befindet sich dort zum aktuellen Zeitpunkt nicht. Damit scheitert ein *chroot* mit diesen Shells.

Das Mastering-System ist nicht eingerichtet, das heißt es gibt keine User, kein Programm ist konfiguriert. Diese Arbeiten werden in »etc« vorgenommen und teilweise ist es günstig, das nach einem *chroot* durchzuführen. Falls Sie also eine externe Shell verwenden, ändern Sie das jetzt mit *chpass* oder einem der anderen Verwaltungstools, kopieren Sie das entsprechende Binär-Package dieser »Lieblingsshell« in die Pseudo-Root des Masters, loggen sich ein und wieder aus und führen Sie dann ein *chroot* dorthin aus:

```
# chroot /pub/ro-image
# pkg_add bash*
# exit
```

Jetzt kann mit *chpass* wieder die reguläre Shell angegeben werden. Nach einem erneuten Ein- und Ausloggen sollte alles wieder passen. Ab sofort funktioniert auch *chroot* ganz normal auf das Mastering-Verzeichnis.

Bevor Sie das tun, sollten Sie aber die Grund-Konfigurationsdateien des Systems umkopieren, das sind vor allem »/etc/wscons.conf«, »/etc/rc.conf« und eventuell noch »/etc/rc.local«.

Dieses System können Sie in einer *chroot*-Umgebung benutzen. Es funktioniert aber nicht mehr, wenn es von einer CD-ROM gestartet wird. Der Grund ist, daß das Betriebssystem in die Verzeichnisse »/etc«, »/dev«, »/tmp«, »/var« und »/root« (und gegebenenfalls auch nach »/home«) schreiben können muß. Diese Möglichkeit wird ihm durch einen Trick gegeben. Es werden mehrere RAM-Disks (mfs-Dateisysteme) beim Systemstart angelegt. Dorthin ist das Schreiben möglich. Diese RAM-Disks benötigen natürlich auch die entsprechenden Dateien. Dafür wird ein Archiv angelegt, in dem sich die entsprechenden Dateien für die RAM-Disks befinden.

Aber der Reihe nach:

Zuerst wird eine neue »fstab« benötigt. In ihr werden die RAM-Disks definiert (Listing A.4).

Auf zwei Spezialitäten muß eingegangen werden:

Das Verzeichnis »/dev« darf nicht einfach als RAM-Disk definiert werden. Es muß ein Union-Mount durchgeführt werden, damit beim Bootvorgang der Zugriff auf die Original-Devices nicht gekappt wird. Außerdem werden keine Device-Nodes definiert. Bei RAM-Disks wird stattdessen der Typ »swap« angegeben.

```
# Device  mnt-point   fstype   size & options                  checks (none)
# ---------------------------------------------------------------------------
swap      /dev        mfs      rw,union,-s=65536                  0      0
swap      /tmp        mfs      rw,nodev,noexec,nosuid,-s=65536     0      0
swap      /var        mfs      rw,-s=131072,noatime                0      0
swap      /etc        mfs      rw,-s=65536                         0      0
swap      /home       mfs      rw,-s=131072                        0      0
swap      /root       mfs      rw,-s=65536                         0      0
kernfs    /kern       kernfs   rw                                  0      0
procfs    /proc       procfs   rw                                  0      0
/dev/fd0a /mnt/floppy1 msdos   rw,noauto                           0      0
```

Listing A.4: Die fstab-Datei für die CD-ROM

In diesem Fall sind außer den normalen Mounts auch noch ein Kern- und und ein Proc-FS definiert. Das müßte nicht sein, schadet aber auch nicht.

Der Parameter *-s* bei den Mounts gibt die Größe der Pseudomedien an. Hier ist es schwierig, eine exakte Aussage über die Idealgröße zu treffen. Das System, das diesem Buch beiliegt, verwendet größere RAM-Disks als in Listing A.2 gezeigt und funktioniert genauso. Allerdings wird es auf Maschinen mit weniger RAM eventuell irgendwann »eng«. Das Problem ist, daß hier Maximalgrößen anzugeben sind und zwar bei NetBSD in Kbyte-Blöcken. Beim Anlegen der RAM-Disks sind diese beileibe nicht so groß. NetBSD hat aber ein Problem mit diesen dynamischen RAM-Disks: Sie werden zwar bei Bedarf dynamisch vergrößert, aber nicht mehr verkleinert, wenn der Speicherplatz nicht mehr benötigt wird. Man kann es also eigentlich nur verkehrt machen. Geht nämlich wegen einer nicht großzügig genug bemessenen RAM-Disk dem System der Platz aus, bleibt es auch stehen.

Sie sollten also auf jeden Fall – allein schon um den Arbeitsspeicher zu schonen – in keinem Fall den *syslogd* auf dem CD-System starten. Schalten Sie ihn am besten in der »rc.conf« mit

```
syslogd=NO
```

ab oder kommentieren Sie in der »etc/syslog.conf« alle nicht lebensnotwendigen Logs aus.

Um die RAM-Disks mit den benötigten Dateien zu bevölkern, muß jetzt noch ein Archiv angelegt werden, das sie enthält. Das Auspacken eines tar.gz-Archivs von der CD ist deshalb erforderlich, weil nach dem Mount der RAM-Disks die Daten in den gemounteten Verzeichnissen (außer »/etc«) nicht mehr sichtbar sind.

Das Anlegen der Datei »mfs.tgz«, die entweder in der CD-Root sein sollte oder aus Gründen der Übersichtlichkeit in einem besonderen Verzeichnis, muß unbedingt vor dem Erzeugen des ISO-Masters erfolgen, da sonst leicht vergessen wird, daß noch Än-

derungen vorgenommen wurden und das Archiv ein älteres etc-Verzeichnis enthält als die CD. Der Master wird später mit dem Programm *mkisofs* geschrieben. In das Skript für den Aufruf von *mkisofs* (*mkrocd*) kommen vorher die Zeilen:

```
#!/bin/sh
cd /pub/ro-master
rm -f etc/*~
rm -f etc/*.bak
tar -cvzf cdboot/mfs.tgz /dev etc var tmp home root
```

Automatisch werden mfs-Laufwerke unter NetBSD nicht gemountet. Außerdem muß im System noch einiges anderes definiert werden. Das ist nicht so ganz einfach. Während die anderen BSD-Systeme eine sehr große, aber einfach strukturierte »rc«-Datei besitzen, verwendet NetBSD ein recht kompliziertes System von Init-Skripten in »/etc/rc.d« für den Bootvorgang.

Welche der Skripten wann gestartet werden, ist einmal durch das System selbst in Form von Abhängigkeitsbedingungen und auch durch die Einträge der Datei »rc.conf« festgelegt.

Ich will hier nicht den Aufbau eines kompletten mehrsprachigen Systems mit einer Vielzahl von Dialogen komplett schildern. Sie können dies anhand der Buch-CD studieren. An dieser Stelle soll das Prinzip anhand von Ausschnitten erläutert werden. Nehmen wir uns zu Beginn den Mountvorgang vor. Das Verzeichnis »etc/rc.d« enthält eine Datei »mountcritlocal«. In diesem Skript werden die kritischen – also lebenswichtigen – Dateisysteme gemountet. Allerdings wird im Skript nur auf eine Routine *mountcritlocal_start* verwiesen. Diese wiederum gibt es gar nicht. Es handelt sich um einen Verweis auf die Routine *mount_critical_filesystems()* in der Datei »etc/rc.subr«. Sie sieht im Original so aus:

```
mount_critical_filesystems()
{
  eval _fslist=\$critical_filesystems_${1}
  for _fs in $_fslist; do
    mount | (
      _ismounted=no
      while read what _on on _type type; do
        if [ $on = $_fs ]; then
          _ismounted=yes
        fi
      done
      if [ $_ismounted = no ]; then
        mount $_fs >/dev/null 2>&1
      fi
    )
  done
}
```

Damit läßt sich leider nicht viel anfangen. Es bleibt nichts anderes übrig, als diese Routine komplett neu aufzubauen. Die Version für das Booten von CD-ROM sieht dann so aus:

```
# mount_critical_filesystems type
#       Go through the list of critical filesystems as provided in
#       the rc.conf(5) variable $critical_filesystems_${type}, checking
#       each one to see if it is mounted, and if it is not, mounting it.
#
# Heavily modified for NetBSD Live!
# Last modifications: 2002/09/22
mount_critical_filesystems()
{
  if [ -f /cdboot/mfs.tgz ] ; then

# Fake the rc script. Otherwise the file systems will be mounted twice:
    if [ "${1}" = "local" ] ; then
      _fslist=$critical_filesystems_beforenet
      for _fs in $_fslist; do
        _ismounted=yes
      done
      # Umount all at the beginning, build the RAM disks and afterwards
      # make all important directories:
      umount -a                 > /dev/null 2>&1
      echo 'Mounting all filesystems...'
      mount -a -t nonfs         > /dev/null 2>&1
      chmod 755 /dev /etc /var  > /dev/null 2>&1
      chmod a+rwx,a+t /tmp      > /dev/null 2>&1
      mkdir /tmp/db             > /dev/null 2>&1
      mkdir /tmp/at             > /dev/null 2>&1
      mkdir /tmp/crash          > /dev/null 2>&1
      mkdir /tmp/cron           > /dev/null 2>&1
      mkdir /tmp/dict           > /dev/null 2>&1
      mkdir /tmp/msgs           > /dev/null 2>&1
      mkdir /tmp/run            > /dev/null 2>&1
      mkdir /tmp/spool          > /dev/null 2>&1
      # if memory file systems mounted, fix them
      # NetBSD's tar has a problem with parameter -C target root
      _olddir=`pwd`
      cd /                      > /dev/null 2>&1
      echo "Fixing up mfs from /cdboot/mfs.tgz"
      # untar the archive with the files of the mfs disks:
      tar -xzf /cdboot/mfs.tgz  > /dev/null 2>&1
      # Changed 2002/09/20. So we will be sure, that all devices
      # are built correctly:
      cd /dev
      ./MAKEDEV all
```

```
    cd /
    echo "Fixing file and directory permissions"
    # Otherwise user 'user' may run in problems:
    chmod 777 /dev/null
    chown -R user:wheel /home/user      > /dev/null 2>&1
    mkdir /home/user/Desktop            > /dev/null 2>&1
    chown -R user:wheel /home/user/.[a-z,A-Z,0-9]* > /dev/null 2>&1
    chown user:wheel /home/user/Desktop > /dev/null 2>&1
    chown -R user:wheel /home/user/.kde/* > /dev/null 2>&1
    chown user:wheel /home/user/.kde/konqueror/bookmarks.xml > \
                                                  /dev/null 2>&1
    # THIS IS NEW IN VERSION 0.5:
    echo ""
    echo -n "Now we try to suck in optional data from "
    echo "a floppy in /dev/fd0a..."
    echo "the file system [[1mmust[[0m be msdos not ffs!"
    mount_msdos /mnt/floppy1 > /dev/null 2>&1
    if [ -d /mnt/floppy1/etc ] ; then
      /bin/cp -R /mnt/floppy1/* / > /dev/null 2>&1
    fi
    umount /mnt/floppy1 > /dev/null 2>&1
    cd ${olddir} > /dev/null 2>&1
    if [ -f /etc/cdconf/useharddisk ] ; then
      if [ ! -f /etc/cdconf/usehdtype ] ; then
        /sbin/mount -t msdos `cat /etc/cdconf/useharddisk` \
                    /mnt > /dev/null 2
      else
        /sbin/mount -t `cat /etc/cdconf/usehdtype` \
                       `cat /etc/cdconf/useharddisk`
      fi
      /bin/cp -R /mnt/etc/* /etc > /dev/null 2>&1
      /sbin/umount /mnt > /dev/null 2>&1
    fi
  else
   # Fake pass 2
    _fslist=$critical_filesystems
    for _fs in $_fslist; do
      _ismounted=yes
    done
  fi
else          # From now on it is the original routine:
  if [ $1 = local ]; then
   _fslist=$critical_filesystems_beforenet
  else
    _fslist=$critical_filesystems
  fi
  for _fs in $_fslist; do
```

```
      mount | (
        _ismounted=no
        while read what _on on _type type; do
          if [ $on = $_fs ]; then
            _ismounted=yes
          fi
        done
        if [ $_ismounted = no ]; then
        mount $_fs >/dev/null 2>&1
        fi
      )
    done
  fi
}
```

Das sieht auf den ersten Blick komplizierter aus als es ist.

Grundsätzlich gilt, daß die Ursprungsroutinen möglichst nicht angefaßt werden sollten, allein schon deshalb, weil sie beim Shutdown auch ausgeführt werden und der soll ja ordnungsgemäß über die Bühne gehen. Deshalb sollte bei zusätzlichen Routinen immer geprüft werden, ob es sich um neue CD-basierte Einträge handelt und nur dann auf die Besonderheiten der CD-Distribution eingegangen werden.

In dieser Routine – auch wenn die große Zahl der Zeilen anderes suggeriert – werden nur die RAM-Disks angelegt und die Daten auf diese RAM-Disks kopiert.

Für ein System, das hartkodiert zwei Netzwerkkarten mit entsprechenden IP-Adressen anspricht und die Daten hin- und herleitet, also einen CD-basierten Router, müssen jetzt nur noch die Netzwerkdateien angelegt werden und – fertig. Ein komfortables Demo-System benötigt mehr. Hier müssen zusätzliche Programme eingerichtet werden, X sollte funktionieren und vieles mehr. Da in NetBSD die »rc«-Datei selbst nicht angetastet werden sollte, gehören weitere Abfragen ebenfalls in die »rc.subr«, außerdem muß eine Datei in »rc.d« dafür sorgen, daß die Routinen auch gestartet werden.

In NetBSD Live! wurde für diese Arbeiten eine Containerfunktion erzeugt. Sie dient der Interaktion der Konfiguration mit dem Benutzer:

```
cdconfig()
{
  # The console behaviour is defined in the file /etc/rc.d/network
  export TERM=vt220

  trap INT TERM
  /usr/bin/tset -sQrm vt220 > /dev/null 2>&1

  # Starting configuration subroutines:
  if [ "`cat /var/use_net`" = "YES" ] ; then
```

```
    SambaCfg              # Samba server configuration subroutine
    DateTimeCfg           # date and time dialog
    XCfg                  # X auto configuration subroutine
    KbdCfg                # Ask user for the keyboard mappings
    rm -f /var/use_net > /dev/null 2>&1
    clear
    Edit_Hosts            # Host name subroutine
  else
    DateTimeCfg           # see above
    XCfg
    KbdCfg
    rm -f /var/use_net > /dev/null 2>&1
    clear
  fi
  gdm=NO                  # switch all display managers off
  kdm=NO
  xdm=NO
  RunDM                   # Run the display manager manually
}
```

Diese Routine wird automatisch in der Datei »rc.d/cdconfig« aufgerufen:

```
#!/bin/sh
#
# $NetBSD: cdconfig,v 1.2.4.1 2002/08/09 19:09:42 jb Exp $
#
# PROVIDE: cdconfig
# REQUIRE: mountcritlocal LOGIN cron inetd

. /etc/rc.subr

name="cdconfig"
start_cmd="cdconfig_start"
stop_cmd=":"
cdconfig_start()
{
  cdconfig
}

load_rc_config $name
run_rc_command "$1"
```

... und damit ist den NetBSD-Bootkonventionen Genüge getan.

Ich will Ihnen hier die einzelnen Unterroutinen nicht alle beschreiben. Sie finden sie ohnehin in der Datei »/etc/rc.subr« auf der CD-ROM, aber eines ist ganz wichtig: Die

Konfiguration verwendet das Programm *dialog* für ansprechend aussehende Dialoge. Das funktioniert nur, wenn vorher eine Konsole definiert wurde. Dazu dienen die Zeilen

```
export TERM=vt220
/usr/bin/tset -sQrm vt220 > /dev/null 2>&1
```

im obigen Skript. Außerdem sollte das Skript nicht abgebrochen werden können. Das verhindert:

```
trap INT TERM
```

Man kann darüber streiten, ob zuerst X und dann die Tastatur konfiguriert werden soll. Auch ob Samba unbedingt gestartet werden sollte. Aber eines muß noch in diesem Zusammenhang erwähnt werden: nicht nur die Mount-Datei wurde geändert, auch die Netzwerk-Datei benötigt einen Eingriff. NetBSD Live! ist nur netzwerkfähig, wenn ein DCHP-Server zur Verfügung steht. Das *dhclient*-Skript sollte aber nicht unnütz aufgerufen werden, da es im Fehlerfall sehr lange versucht, einen Server zu finden. Deshalb wird vorab in der Datei »/etc/rc.d/network« gefragt, ob Netzwerk via DHCP überhaupt verwendet werden soll:

```
network_start()
{
# This routine is heavily modified for NetBSD Live!
  local tst

  trap INT TERM

  if [ -f /cdboot/mfs.tgz ] ; then

    export TERM=vt220
    /usr/bin/tset -sQrm vt220 > /dev/null 2>&1

    /usr/pkg/bin/dialog --backtitle "Boot Configuration" \
                        --title "[ Information ]" \
                        --msgbox "You may change some defaults of \
this distribution for fine tuning different parts of NetBSD." 6 72
    /usr/pkg/bin/dialog --backtitle "Boot Configuration" \
                        --title "[ Main Config
                        --yesno "Use network (dhcp server required) ?" \
                        6 72
    tst=$?

    if [ ${tst} = 0 ] ; then
      USE_NET=YES
    else
```

```
    USE_NET=NO
    dhclient=NO
    echo ""            >> /etc/rc.conf
    echo "dhclient=NO" >> /etc/rc.conf
    hostname=`cat /etc/myname`
    hostname $hostname
    domainname=`cat /etc/defaultdomain`
    domainname $domainname
  fi

  echo ${USE_NET} > /var/use_net

  if [ "${USE_NET}" = "YES" ] ; then
    if [ ! -f /etc/cdconf/name ] ; then
      if /usr/pkg/bin/dialog --backtitle "Boot Configuration" \
                             --title "[ Main configuration ]" \
                             --inputbox 'Change host name:' 8 66 \
                             `cat /etc/myname ; then
        hostname=`cat /etc/myname`
        if [ -z "$hostname" ] ; then
          # Write back defaults into the file again
          echo "cdrom" > /etc/myname
          hostname=`cat /etc/myname`
        else
          hostname $hostname
          export HOSTNAME=$hostname
          echo $hostname > /etc/smbname
          touch /etc/cdconf/name > /dev/null 2>&1
        fi
      fi

      if /usr/pkg/bin/dialog --backtitle "Boot Configuration" \
                             --title "[ Main configuration ]" \
                             --inputbox "Correct NIS domainname:" \
                             8 66 `cat /etc/defaultdomain` \
                             2> /etc/defaultdomain; then
        domain=`cat /etc/defaultdomain`
        if [ -z "$domain" ] ; then
          echo "mydomain" > /etc/defaultdomain
          domain=`cat /etc/defaultdomain`
        else
          domainname `cat /etc/defaultdomain`
          domain=`cat /etc/defaultdomain`
          touch /etc/cdconf/name > /dev/null 2>&1
        fi
      fi
    fi
```

```
    # end of no name file found
  fi
  # Ab hier folgen die Originalroutinen
fi
....
```

Hiermit ist die Beschreibung des »rc«-Systems abgeschlossen. Eines fehlt und ist auch bei dieser Version nicht gelöst: die Uhrzeit. Während Open- und FreeBSD und auch NetBSD in der Version 1.5.2 die Uhrzeit nicht aus dem BIOS übernahmen wenn sie von einer CD booteten, macht das NetBSD 1.6[1] jetzt richtig. Allerdings nur für die in der Datei »/etc/localtime« eingestellte Zeitzone und nur, wenn das BIOS der Maschine auf UTC gestellt ist. Man könnte jetzt beim Hochfahren des Systems die Zeitzone abfragen und die entsprechende Zeitzonendatei nach »/etc« kopieren. Diese Version der CD-Distribution macht das aber nicht und geht einfach von der mitteleuropäischen Zeit aus.

Jetzt muß noch X zum Laufen gebracht werden. Das ist genauso kritisch und hat beim Umstieg von NetBSD 1.5.2 nach 1.6 einiges an Kopfzerbrechen bereitet.

A.2.3 X von CD

Ein ordentliches Demo-System benötigt die Möglichkeit, die grafische Oberfläche zu zeigen. Vorab muß deshalb dafür gesorgt werden, daß der X-Server gewechselt werden und der Tastaturcompiler schreiben kann.

Wenn Sie noch nichts vom Tastaturcompiler gehört haben, ist das nicht weiter schlimm. Erstens ist der recht versteckt, macht etwas, was er nicht tun soll und ohne ihn können Sie immerhin noch mit der US-Tastatur arbeiten.

Die Tastaturbelegung von X wird nicht – wie man meinen sollte – bei der Definition der Tastatur im X-Konfigurationsprogramm fest verdrahtet. Bei jedem Start der grafischen Oberfläche wird vom Compiler */usr/X11R6/lib/X11/xkb/xkbcomp* dynamisch im Verzeichnis »/usr/X11R6/lib/X11/xkb/compiled« die Tastaturbelegungstabelle erzeugt. Sie könnten jetzt einwenden, daß dem eigentlich nicht so sein sollte, da prinzipiell dieser Verzeichnisbaum auch per NFS oder gar read-only gemountet werden können sollte. Damit haben Sie auch völlig recht. Aber es wird halt so gemacht und zwar sowohl bei XFree86 3.3.x als auch bei 4.2.x. Auf der CD-ROM wird dieses Verzeichnis aber in der Tat schreibgeschützt eingebunden, was dazu führt, daß unter X eigentlich nur eine amerikanische Tastaturbelegung möglich ist. Abhilfe schafft ein Link. Dazu wird in »/var« der CD ein Verzeichnis »compiled« angelegt, das ursprüngliche Verzeichnis »/usr/X11R6/lib/X11/xkb/compiled« gelöscht und durch einen symbolischen Link ersetzt:

---

[1] Hat sich da jemand bei der NetBSD-Gruppe über die Uhrstellerei in der NetBSD-1.5.2-basierten NetBSD Live! 0.4 geärgert und den Fehler im Kernel beseitigt?

```
cd usr/X11R6/lib/X11/xkb/
rm compiled
ln -s ../../../../../var/compiled compiled
```

Damit funktioniert die Tastatur wie gewünscht. Ähnliches muß für den X-Server gemacht werden. Unter XFree86 4.2.0 gibt es zwar nur noch einen, wenn man alternativ aber einen der alten Server von XFree 3.3.6 einsetzen können will, muß das sein:

```
cd var
ln -s ../usr/X11R6/bin/XFree86 X
cd ../usr/X11R6/bin
rm X
ln -s ../../../var/X X
```

Mit diesem doppelten Verlinken kann durch Austausch des Links »X« in »/var« der CD-ROM jederzeit ein anderer X-Server, beispielsweise durch ein

```
ln -s usr/X11R6/X11R6.3/bin/XF86_SVGA var/X
```

der XFree86-3-SVGA-Server eingesetzt werden.

Womit zwei Probleme, aber nicht die hauptsächlichen, gelöst wären. Denn jetzt laufen zwar die X-Server, es fehlt aber die X-Konfiguration. Es kann nicht sein, daß bei jedem Systemstart *xf86cfg* oder noch schlimmer *xf86config* aufgerufen werden muß. Die automatische Konfiguration mit einem Aufruf von *XFree86 -configure* bringt derart magere Resultate, daß man es eigentlich vergessen kann. Hinzu kommt, daß die CD möglichst flexibel sein soll und die Konfigurationsdateien von XFree86 3 und 4 nicht in geringster Weise kompatibel sind.

Für XFree86 3.3.6 wurde die Brechstangenmethode gewählt. Beim Systemstart kann aus einer Reihe von Servern mit einigen Optionen ausgewählt werden. Das funktioniert auf vielen Systemen sehr gut. Wenn alle Stricke reißen, wählt man den VGA-Server. Er funktioniert zwar nicht schön, aber läßt sich eigentlich immer einsetzen.

Das Problem mit XFree86 4 wurde durch eine Anleihe bei Knoppix[1] gelöst. Hier wird die Grafik durch die Kombination eines Skripts, des Hardware-Konfigurationsprogramms und einer Input-Datei erzeugt. Da das Hardware-Erkennungstool nicht zur Verfügung steht, ist eine Grafikkartenerkennung direkt nicht möglich. Deshalb verwendet X den VESA-Modus der Grafikkarte. In den meisten Fällen funktioniert das, insbesondere bei modernen Karten. Mit älteren Grafikkarten verwendet man dann einen XFree86-3-Server.

---

[1] Knoppix (*www.knoppix.net*) ist eine ausgereifte CD-basierte Linux-Version auf Debian/GNU Linux basierend. Aber es ist eben Linux und kein BSD!

Das Skript befindet sich in »/etc/X11« auf der CD-ROM und wurde nur unmaßgeblich gegenüber der Originalversion verändert.

Seit die CD erstellt wurde, habe ich am Skript noch kleinere Änderungen vorgenommen. Eine rudimentäre Hardwareerkennung wurde mit den folgenden Zeilen realisiert:

```
# New for NetBSD configuration
/usr/X11R6/bin/XFree86 -configure 2> /dev/null
echo -n "XMODULE=" > /etc/X11/xserver
grep "BoardName" /root/XF86Config.new | cut -d '"' -f 2 | \
     cut -d ' ' -f 1 >> /etc/X11/xserver
echo -n "XDESC=" >> /etc/X11/xserver
grep "BoardName" /root/XF86Config.new | cut -d '"' -f 2 | \
     cut -d ' ' -f 1 >> /etc/X11/xserver

## Continue with the original script:
# Read in what hwsetup has found for X
[ -f /etc/X11/xserver ] && . /etc/X11/xserver
```

Es scheint zu funktionieren. Sie können die Zeilen – wenn Sie eine eigene CD-Distribution zusammenstellen – in Ihre Version von »etc/X11/mkxf86config« einbauen.

Wichtig ist, daß in »/etc/X11« keine vorbereitete Datei »XF86Config« sein darf. Sonst funktioniert das Knoppix-Skript nicht.

Damit ist die Grund-CD aufgebaut.

A.2.4 Anwendungen

Bis jetzt wurde die Grundkonfiguration der CD-Distribution beschrieben. Aber es sind – außer eventuell einer Shell – keinerlei weitere Anwendungen installiert. Diese sollten nicht aus dem installierten Grundsystem genommen, sondern aus Binär-Packages neu installiert werden. Ideal ist es, wenn Sie die Binär-Packages von einer CD-ROM nehmen können, dann ist es sehr einfach:

Legen Sie auf dem CD-Master das Mount-Verzeichnis »/mnt« an (Sie brauchen es ohnehin). Führen Sie – wenn nicht bereits erfolgt – ein *chroot* auf das Masteringverzeichnis aus und mounten Sie die CD-ROM:

```
chroot /pub/ro-image
mount_cd9660 /dev/cd0d /mnt/cdrom
```

Wechseln Sie in das Verzeichnis mit den Packages und installieren Sie die gewünschten Programme. Installieren Sie nicht einfach drauflos, sondern sehr überlegt und überprüfen Sie hin und wieder mit

```
# du -m /
```

daß der Master nicht merklich größer als 700 Mbyte wird. Ein moderner Rohling faßt maximal 702 Mbyte und wir können vom Master nach den Installationen auch wieder einiges löschen. Aber installieren Sie Anwendungen unbedingt mit Bedacht. Ein KDE-3-Komplettsystem bringt das System schon nahezu ans Limit!

Sind die Installationen abgeschlossen, geht es ans Löschen. Entfernen Sie alle Include-Verzeichnisse, sie werden nicht benötigt und kosten in »/usr/pkg« und »/usr/X11R6« doch einiges an Platz. Gehen Sie auch ruhig alle Verzeichnisse durch und löschen Sie in den share-Unterverzeichnissen nicht benötigte Locales.

Zum Schluß sollten Sie – falls es nichts mehr zu installieren gibt – auch ganz radikal den gesamten Verzeichnisbaum »var/db/pkg« löschen.

Aber achten Sie immer darauf, daß Sie bei diesen Arbeiten sich auch wirklich auf dem CD-Master befinden und keineswegs die Verzeichnisse der normalen Installation löschen. Das hat verheerende Auswirkungen!

Ist alles fertig, sollten Sie den X-Server einrichten und in der *chroot*-Umgebung einmal ganz normal arbeiten. Rufen Sie alle Programme auf, damit diese ihre Konfigurationsdateien in den Heimatverzeichnissen ablegen können. Richten Sie in Ruhe die grafische Oberfläche, den Display-Manager und so weiter ein.

Sind Sie mit dem System zufrieden, können Sie sich daranmachen, es das erste Mal auf eine CD zu schreiben. Es wird nicht der einzige Rohling sein, den Sie benötigen. Die Verwendung einer CD-R/W ist – falls das System nicht zu groß für eine CD-R/W ist – durchaus empfehlenswert.

Jetzt muß das System mit *mkisofs* gemastert und dann gebrannt werden. Das Skript für das Mastering sollte ungefähr so aussehen:

```
#!/bin/sh
cd /pub/ro-image          # ... oder wo Sie Ihren Master angelegt haben
dir=`pwd`

# Einstellen der Defaults:
cd etc
rm *core > /dev/null 2>&1
rm *~ > /dev/null 2>&1
echo cdrom > myname
echo mydomain > defaultdomain
cp XF86Config.vga XF86Config
cd ../var
rm X > /dev/null 2>&1
ln -s ../usr/X11R6/bin/XFree86 X

cd ../root
rm .bash_history > /dev/null 2>&1
rm *core > /dev/null 2>&1
rm *~ > /dev/null 2>&1
```

```
cd ..

cd home/user
rm .bash_history > /dev/null 2>&1
rm *core > /dev/null 2>&1
cd ../..
tar -cvzf cdboot/mfs.tgz /dev etc var tmp home root kern

# Do not use option -r. This destroys the suid bits!
mkisofs -A "NetBSD-Readonly-System v0.5" -b "cdboot/boot-big1.fs"
        -hide-rr-moved -c "cdboot/boot.cat" -R -d -D -L \
        -P "Hier tragen Sie Ihren Namen ein <... Und Ihr E-Mail>" \
        -p "Hier das selbe wie in -p" -V "NetBSD Live! 0.5" \
        -o "/pub/isoimages/NetBSDLive.iso" ${dir}
```

Diese CD besitzt auf einem Windows-System keine langen Dateinamen. Aus Platzgründen sind die Joliet-Extensions weggelassen. Sie können Sie, wenn genug Platz sein sollte, mit dem zusätzlichen Parameter *-J* einschalten.

Das ISO-Image wird in »/pub/isoimages« angelegt. Diesen Pfad müssen Sie auch anpassen.

Wurde das Image angelegt und ist es unter 700 Mbyte groß, können Sie entweder noch ein paar Programme installieren oder alles belassen und das ISO-Image mit einem Brennprogramm auf einen Rohling übertragen.

Wenn Sie den Rechner neu booten und im BIOS die Bootreihenfolge ändern, sollte er direkt von der CD starten.

A.2.5 Mastern und brennen von CDs

Images anlegen mit »mkisofs«

Unter allen Unix-Derivaten werden CD-ISO-Images mit dem Programm *mkisofs* oder seinem Geschwister *mkhybrid* erzeugt. Liest man im Internet Dokumentationen zum Mastern und Brennen von CDs durch, stellt man fest, daß dieses Kommandozeilenprogramm die größte Hürde für das Brennen darzustellen scheint, weil die Autoren lange Exkurse über das Austüfteln der notwendigen Parameter schreiben.

Hat man die Struktur der Kommandozeilenparameter einmal verstanden, kann man auf Interface-GUI-Tools verzichten. Ein kleines Shell-Skript, bei dem nur einige der Parameter für andere Images angepaßt werden, reicht dann aus.

Das Programm *mkisofs* (das gilt genauso über das alternative *mkhybrid*) läßt sich über eine Konfigurationsdatei ».mkisofsrc« im Heimatverzeichnis konfigurieren. Davon ist aber abzuraten, um die Konfiguration beim Brennen transparent zu halten. Auf der Kommandozeile angegebene Parameter überschreiben zwar die der Konfigurationsdatei, trotzdem taucht bei der Verwendung der Konfigurationsdatei recht schnell das Problem auf, daß beim ISO-Erzeugen unter einem anderen User-Account auf einmal etwas nicht

mehr stimmt, weil hier die Steuerdatei fehlt. Dieses Problem wäre bei kompletter Angabe der Parameter auf der Kommandozeile nicht aufgetreten.

An dieser Stelle sollten keineswegs alle möglichen Parameter des ISO-Schreibprogramms beschrieben werden. Die Manpages sind sehr lang und die Parameter sind vielfältig. Hier geht es nur um die Grundlage, ein normales bootfähiges ISO-Image zu schreiben.

Die Grundparameter

In Abschnitt A.2.4 dieses Anhangs wurde Ihnen die Kommandozeile für *mkisofs* ohne Erläuterung einfach vorgegeben:

```
# mkisofs -A "NetBSD-Readonly-System v0.5" \
          -b "cdboot/boot-big1.fs" \
          -hide-rr-moved \
          -c "cdboot/boot.cat"
          -R -d -D -L \
          -P "Hier tragen Sie Ihren Namen ein <... Und Ihr E-Mail>" \
          -p "Hier das selbe wie in -p" \
          -V "NetBSD Live! 0.5" \
          -o "/pub/isoimages/NetBSDLive.iso" /pub/ro-image
```

Hier zuerst die Erläuterung der zeichenkettenorientieren Parameter:

-A	Der Name der CD-ROM. Dieser Name ist normalerweise nicht zu sehen. Er kann nur mit speziellen Low-Level-Werkzeugen abgefragt werden. Der Parameter *-A* steht für »Application«. Der Name darf 128 Zeichen lang sein. Wenn Sie eine ».mkisofsrc« verwenden, ist er hier durch die Kennung *APPI=Name* festzulegen.
-V	Das ist das Volume-Label. Dabei handelt es sich um den Verzeichniseintrag, über den der Datenträger identifiziert wird. In der ».mkisofsrc« wird die Kennung über *VOLI=Name* festgelegt. Unter Unix ist das Volume Label eigentlich überflüssig. Die Kennung wird unter Windows und auf dem Macintosh verwendet.
-P	In der ».mkisofsrc« würde dieser Eintrag durch die Kennung *PUBL=Name* festgelegt. Es handelt sich um den Namen des Herausgebers der CD. Wenn Sie die CD für eine Firma erstellen, muß deren Name und E-Mail-Adresse in diesem Feld aufgeführt sein. Das Feld darf eine maximale Länge von 128 Zeichen besitzen.
-p	Hierhin gehört der Name des Masterers (»Preparer«) der CD-ROM. In der ».mkisofsrc« würde dieser Eintrag durch die Kennung *PREP=Name* festgelegt. Außer dem Namen sollte auch noch die E-Mail-Adresse angegeben sein. Wie bei allen Stringangaben ist die Länge auf 128 Zeichen begrenzt.

-o	Hier werden der Pfad zum und der Name des ISO-Images angegeben. Der Pfad muß vorhanden sein. Ein bereits mit dem selben Namen angelegtes ISO-Image wird ohne Nachfrage überschrieben. *mkisofs* prüft nicht, ob für das Ziel-ISO-Image genug Platz vorhanden ist. Falls der Platz nicht ausreicht, wird während des Schreibvorgangs abgebrochen, das bereits angelegte Image aber nicht gelöscht. Es ist dann aber nicht funktionsfähig.

Die Erzeugung des ISO-Images wird mit den folgenden Parametern gesteuert:

-R	Erzeugen der SUSP- und RR-Records für RockRidge. Dieser Parameter ist Pflicht.
-d	Belassen von Punkten an Dot-Dateien. Der ISO-Standard erlaubt nicht, daß der Punkt in Datei- und Verzeichnisnamen vorkommt, außer als Trenner von Dateiname und -Endung. Wird der Parameter *-d* nicht angegeben, wird aus einer Datei ».profile« eine Datei mit dem Namen »_profile« erzeugt. Das ist nicht gewünscht.
-D	Eine Relozierung der Pfade soll nicht erfolgen. Die Dateien sollen so auf die CD übernommen werden wie sie sind. Hierbei wird ein Verzeichnis »/RR_MOVED« erzeugt, das mit dem Parameter *-hide-rr-moved* in ».rr_moved« umbenannt, also versteckt wird.
-L	Hiermit wird zugelassen, daß auch Dateien im ISO-Verzeichnis mit einem Punkt beginnen. Das verstößt zwar gegen die ISO-Norm, erhöht aber die Übersichtlichkeit.

Beim Brennen von CDs wird gern der zusätzliche Parameter *-r* verwendet. Damit werden die Benutzerrechte auf durchschnittliche Werte gesetzt, also alle unter einem normalen Benutzeraccount angelegten Dateien *root* zugewiesen. Das ist bei einer Installations-CD sehr sinnvoll, bei einer CD, auf der User angelegt sind, die von der CD arbeiten, aber nicht. Setzt man den Parameter *-r*, können auf dem CD-System angelegte User beispielsweise die grafische Oberfläche nicht starten, weil die Benutzerrechte und die Dateiattribute nicht mehr stimmen.

Absolut problemlos ist die Verwendung des Parameters *-J*. Er legt fest, daß außer langen Dateinamen im RockRidge-Format auch solche im Joliet-Format von Windows auf der CD angelegt werden (»Joliet Directories«). Diese Erweiterungen kosten allerdings Speicherplatz auf der CD. Sie sollten also nur verwendet werden, wenn noch einige Mbyte freier Speicher verfügbar sind. Der Vorteil der Joliet-Verzeichnisse ist, daß die Daten und Verzeichnisse der CD auch auf anderen Plattformen – im Notfall mit besonderen Treibern – gelesen werden können. Joliet-Treiber gibt es außer für Windows (wo sie im System automatisch enthalten sind) auch für MacOS ab Version 7 und für OS/2. Joliet-CDs werden heute auch von Linux, BSD und anderen Unix-Derivaten verstanden.

Joliet kennt keine Unix-Links und ist deshalb bei echten Unix-CDs nur eine Krücke[1].

-T	Dieser Parameter ist im obigen Aufruf ebenfalls nicht enthalten. Hiermit werden in jedem ISO-Verzeichnis Übersetzungstabellen angelegt, die auf Systemen ohne lange Dateinamen deren Restaurierung ermöglichen. Naturgemäß kostet das Anlegen einer Umsetzungstabelle in jedem Verzeichnis aber zusätzlichen Platz. Es ist deshalb nur sinnvoll, wenn noch genug Platz auf dem zu erzeugenden Datenträger zur Verfügung steht.

Die weiteren Parameter dienen der Festlegung des El-Torito-Bootimages. Das El-Torito-Format definiert ein Diskettenimage, das im Format 1,2 Mbyte, 1,44 Mbyte oder 2,88 Mbyte vorliegen muß, von dem das System starten kann. Bei Vorfinden eines solchen Images auf dem Datenträger und entsprechender Einstellung im BIOS wird versucht, ein Betriebssystem von dieser virtuellen Diskette zu starten. Das Originalformat von El-Torito erlaubt die Angabe genau eines solchen Images, dessen *relativer* Pfad hinter dem Parameter *-b* angegeben werden muß. Außerdem ist zwingend die zusätzliche Angabe eines Boot-Katalogs vorgeschrieben. Hinter dem Parameter *-c* muß ebenfalls der *relative* Pfad zu dieser Datei angegeben sein. *Relativ* heißt in beiden Fällen, daß sich das Image und die Datei auf dem Master befinden müssen. Befindet sich beispielsweise die Datei im Verzeichnis »/pub/ro-image/cdboot« und heißt »boot-big1.fs« und ist als Ausgangsverzeichnis für das Mastering »/pub/ro-image« angegeben, ist als Parameter *-b cdboot/boot-big1.fs* anzugeben.

Neuere Versionen von mkisofs erlauben die Angabe von bis zu 63 alternativen Boot-Images. Selbstverständlich müssen sich diese alle auf dem Master befinden. Der Aufruf für das alternative Booten muß folgendermaßen aussehen:

```
-eltorito-boot cdboot/boot-big1.fs \
-eltorito-alt-boot -eltorito-boot cdboot/boot-laptop.fs \
...
```

Zusätzlich kann noch bei jedem Image der Parameter *-hide* angegeben werden, beispielsweise mit *-hide cdboot/boot-big1.fs*. Es ist bei mehreren Images anstelle von *-b image* jeweils die Langfassung des Befehls anzugeben (*-eltorito-boot*). Vor *jedem* der zusätzlichen Boot-Images ist außerdem der Zusatzbefehl *-eltorito-alt-boot* anzugeben.

-eltorito-alt-boot führt dazu, daß bei entsprechender BIOS-Unterstützung beim Start des Systems zum einem Tastendruck aufgefordert wird, falls ein anderes als das erste Diskettenimage gebootet werden soll. Leider unterstützen viele PC-BIOS-Versionen dies aber nicht.

[1] Es gibt sogar kommerzielle Hersteller von Linux-Software, die auf die Vorteile des RockRidge-Formats verzichten und Joliet-CDs ausliefern. Diese CDs sind dann wohl unter Windows gebrannt worden.

So enthält die dem Buch beigelegte Version von NetBSD Live! zwei Boot-Images, eines für normale Maschinen, eines für Laptops, die mit dem generischen Kernel nicht zurechtkommen. Ausgerechnet IBM-Thinkpads, die diesen alternativen Kernel benötigen, besitzen aber keine Unterstützung für alternative Boot-Disketten.

```
MKISOFS(8)                                                    MKISOFS(8)

NAME
       mkisofs  -  create an hybrid ISO9660/JOLIET/HFS filesystem
       with optional Rock Ridge attributes.

SYNOPSIS
       mkisofs [ -abstract FILE ] [ -allow-lowercase ] [  -allow-
       multidot ] [ -biblio FILE ] [ -cache-inodes ] [ -no-cache-
       inodes ] [ -b eltorito_boot_image ] [ -eltorito-alt-boot ]
       [  -B  sparc_boot_image_list ] [ -G generic_boot_image ] [
       -gui ] [ -C #,# ] [ -hard-disk-boot ] [ -no-emul-boot ]  [
       -no-boot ] [ -boot-load-seg ] [ -boot-load-size ] [ -boot-
       info-table ] [ -c boot_catalog ]  [  -check-oldnames  ]  [
       -check-session  FILE  ]  [ -copyright FILE ] [ -A applica-
       tion_id ] [ -f ] [ -d ] [ -D ] [ -dir-mode mode ] [ -File-
       mode  mode ] [ -gid gid ] [ -hide glob ] [ -hide-list file
       ] [ -hidden glob ] [ -hidden-list file  ]  [  -hide-joliet
       glob ] [ -hide-joliet-list file ] [ -hide-joliet-trans-tbl
       ] [ -hide-rr-moved ] [ -iso-level level ] [ -input-charset
       charset  ]  [ -output-charset charset ] [ -J ] [ -jcharset
       charset ] [ -l ] [ -L ] [ -log-file  log_file  ]  [  -max-
       iso9660-filenames  ]  [ -M path | device ] [ -new-dir-mode
       mode ] [ -nobak ] [ -no-bak ]  [  -no-split-symlink-compo-
       nents  ] [ -no-split-symlink-fields ] [ -pad ] [ -no-pad ]
       [ -path-list file ] [ -p preparer ] [ -print-size ]  [  -P
       publisher  ] [ -quiet ] [ -r ] [ -R ] [ -relaxed-filenames
       ] [ -sort sort_file ] [ -sysid ID ]  [  -T  |  -table-name
       TABLE_NAME  ]  [  -ucs-level  level ] [ -uid uid ] [ -use-
       fileversion ] [ -U ] [ -no-iso-translate ] [  -v  ]  [  -V
       volid  ] [ -volset ID ] [ -volset-size # ] [ -volset-seqno
       # ] [ -x path ] [ -exclude-list file ] [ -z ] [ -m glob  ]
       [  -hfs  |  -apple  ]  [  -map  mapping_file  ]  [  -magic
       magic_file ] [ -probe ] [ -no-desktop ] [  -mac-name  ]  [
       -boot-hfs-file    driver_file    [    -part    ]    [    -auto
       AutoStart_file ] [ -cluster-size size ] [ -hide-hfs glob ]
       [  -hide-hfs-list file ] [ -hfs-volid hfs_volid ] [ -icon-
       position ] [ -root-info FILE [ -prep-boot FILE ] [  -hfs-
       creator  CREATOR ] [ -hfs-type TYPE ] [ -input-hfs-charset
       charset ] [ -output-hfs-charset charset ] [ -hfs-unlock  ]
       [  -hfs-bless  folder_name  ]  [  --cap ] [ --netatalk ] [
       --double ] [ --ethershare ] [ --ushare ] [ --exchange ]  [
       --sgi ] [ --xinet ] [ --macbin ] [ --single ] [ --dave ] [
       --sfm ] -o filename pathspec [pathspec]

DESCRIPTION
       mkisofs is effectively a pre-mastering program to generate
```

Abb. A.2: Die Manpage mit der Parameterliste von mkisofs

Nur der Vollständigkeit halber soll hier noch auf eine weitere Bootmöglichkeit eingegangen werden. Neue Versionen von *mkisofs* (auch die bei NetBSD 1.6 in den Packages befindliche) unterstützen zusätzlich zum Booten von Diskettenimages auch das Booten von Festplatten-Images. Diese Art des Boots findet beispielsweise bei der Installation von Darwin 6.0.2 Verwendung. Dazu muß das Image aber Eins-zu-Eins einer Festplatte entsprechen und dabei sogar einen Master-Boot-Record besitzen. Die passende Parameterliste für das Image von Darwin ist beispielsweise

```
-hard-disk-boot -b "usr/standalone/i386/cdboot.dmg" \
-c "usr/standalone/i386/boot.cat" \
```

Bei der Angabe von *-hard-disk-boot* gelten ansonsten die selben Grundlagen wie bei Disketten-Images. Diese neue Art des Bootens hat aber den Vorteil, daß die Größenbeschränkung von 2,88 Mbyte aufgehoben ist.

Als letzter Parameter bei *mkisofs* ist immer der Pfad zum Hauptverzeichnis des Masters anzugeben. Es dürfen keine weiteren Parameter mehr folgen.

Wenn Sie die Manpage von *mkisofs* aufrufen, werden Sie noch zahlreiche weitere mögliche Parameter sehen. Im normalen Einsatz brauchen Sie sie nicht. Sie sollten sich also nicht verwirren lassen. Ohnehin sind etwa die Hälfte der Parameter für das Erzeugen von Hybrid-CDs, die zusätzlich zum ISO-Dateisystem auch das HFS-Dateisystem von MacOS enthalten, zuständig. Als normaler Anwender kommt man mit diesen Optionen nicht in Berührung.

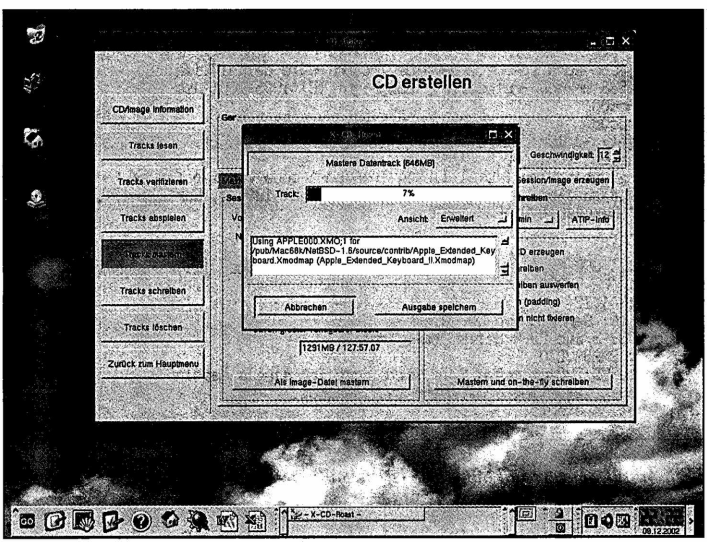


Abb. A.3: Wer sich nicht mit Kommandozeilenparameter »belasten« will verwendet X-CD-Roast als Frontend auch für mkisofs

Das eigentliche Brennen von CDs

Das Brennen von CDs ist unter NetBSD sehr viel einfacher gelöst als bei anderen Betriebssystemen. Durch den Device-Aufbau mit dem Gerät »/dev/(r)/cdXa« für SCSI- und ATAPI-Medien kann das Programm *cdrecord,* das eigentlich für SCSI-Medien ausgelegt ist, auch für ATAPI-Medien verwendet werden.

mkisofs legt ein ISO-Image an. Dies ist ein direktes Image einer CD. Sie erzeugen solch ein ISO-Image unter NetBSD genauso wie von einer vorhandenen CD-ROM, CD-R oder CD-RW mit dem *dd*-Befehl, indem Sie den folgenden Aufruf tätigen:

```
# dd if=/dev/rcd0a of=filename.iso bs=2k1
358274+0 records in
358274+0 records out
733745152 bytes transferred in 639.728 secs (1146964 bytes/sec)
```

Der Parameter */dev/rcd0a* ist die Kennung für das »Raw-Device«. Das bedeutet, daß nicht die eigentlichen Daten gelesen werden, sondern direkt die Rohdaten aus dem Device. Die Angabe von *bs=2k* teilt *dd* die Größe von zwei Kbyte als Blockgröße (bs = block size) mit.

Egal, wie Sie Ihr ISO-Image erzeugt haben: es muß auf einen leeren Rohling, entweder auf eine CD-R oder eine CD-RW passen.

```
cdrecord: Invalid argument. Cannot open SCSI driver.
cdrecord: For possible targets try 'cdrecord -scanbus'. Make sure you are root.
root@desktop
[/pub/isoimages]#cdrecord speed=12 dev=/dev/rcd0d -data delphi7.iso
Cdrecord 1.10 (i386-unknown-netbsd1.6.) Copyright (C) 1995-2001 Jso
scsidev: '/dev/rcd0d'
devname: '/dev/rcd0d'
scsibus: -2 target: -2 lun: -2
Using libscg version 'schily-0.5'
cdrecord: Warning: using inofficial libscg transport code version (frueauf-scsi-
bsd.c-1.40nb1 '@(#)scsi-bsd.c   1.40 01/03/18 Copyright 1997 J. Schilling').
Device type    : Removable CD-ROM
Version        : 0
Response Format: 1
Vendor_info    : '_NEC    '
Identifikation : 'NR-7900A         '
Revision       : '1.22'
Device seems to be: Generic mmc CD-RW.
Using generic SCSI-3/mmc CD-R driver (mmc_cdr).
Driver flags   : SWABAUDIO
Starting to write CD/DVD at speed 8 in write mode for single session.
Last chance to quit, starting real write in 7 seconds.
0 seconds. Operation starts.
```

Abb. A.4: Ein Aufruf von cdrecord auf eine ATAPI-Laufwerk, hier von NEC

Eigentlich ist es egal, mit welchem Programm und unter welchem Betriebssystem so ein Image gebrannt wird. Nehmen wir einmal an, der NetBSD-Rechner hat keinen Brenner eingebaut. Eine FreeBSD-4-Maschine im Netz besitzt aber einen ATAPI-Brenner. Dann übertragen Sie das Image und brennen es mit *burncd*. Die Parameter lauten:

```
# burncd -f /dev/acd0c -s 12 data data.iso fixate
```

[1] CD-ROMs im ISO9660-Format, die kein Disklabel besitzen, werden sonst immer über die Kennung »d« für »Disk« angesprochen. Beim Verwenden des Raw-Devices bei CDs (und auch bei FAT-Disketten) wird das Gerät über »a« eingebunden, obwohl beide naturgemäß kein Disklabel besitzen. Noch seltsamer ist, daß bei Angabe von »/dev/rcd0d« das Image genauso angelegt wird, aber zum Schluß die Meldung *dd: /dev/rcd0d: Invalid argument* erscheint.

Die Parameter bedeuten, daß auf das erste ATAPI-Device mit zwölffacher Schreibgeschwindigkeit das Datenimage »data.iso« gebrannt und dann fixiert wird.

Wenn Sie das ISO-Image unter Windows brennen, ist die Vorgehensweise abhängig vom Brennprogramm. Bei Nero muß die Datei in ».nrg« umbenannt und dann als Fremddatei verwendet werden. Die von Nero angebotenen Parameter für die Einbindung können bei den Voreinstellungen belassen werden. Bei Easy CD Creator und bei WinOnCD ist weiter nichts zu beachten. Beide verwenden selbst reine ISO-Images und übernehmen die Datei anstandslos.

Um die CD-R unter NetBSD zu beschreiben, verwendet man entweder das Kommandozeilenprogramm *cdrecord* selbst oder eines der über die Packages installierbaren GUI-Interfaces hierfür.

cdrecord befindet sich im gleichnamigen Package unter der Rubrik »sysutils«, die diversen Interfaces wie *gnometoaster*, *xcdroast* oder *kconcd* (nur mit KDE 1.1.2) erleichtern die Übergabe der Parameter an das Programm. Prinzipiell mündet alles aber in einem Aufruf von

```
# cdrecord speed=12 dev=/dev/rcd0d -data data.iso
```

Beachten Sie, daß *dd* aus »rcd0a« liest, *cdrecord* aber auf »rcd0d« schreibt.

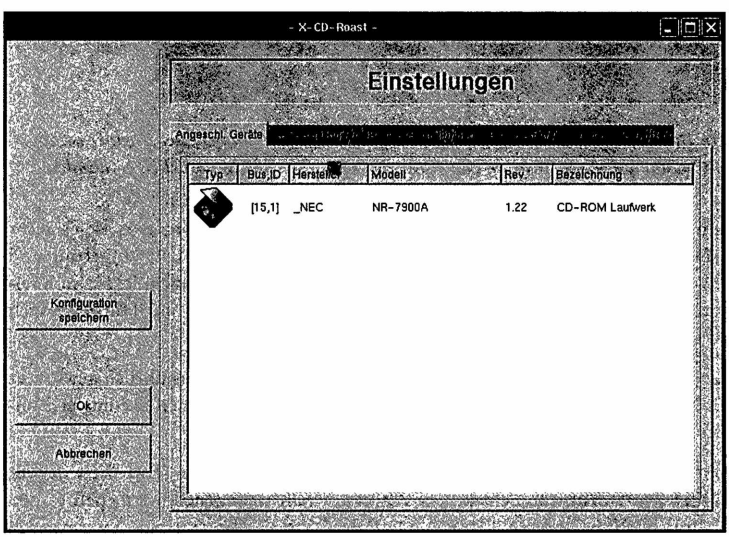


Abb. A.5: X-CD-Roast erkennt auch ATAPI-Brenner zuverlässig

Von den X-Interfaces funktioniert X-CD-Roast (*xcdroast*) am besten. Es erkennt auf dem Intel-Ports zuverlässig auch ATAPI-Brenner (Abb. A.5). Auf dem Macintosh mit 68040-Prozesor meldete das Programm zuerst, es daß keine SCSI-Geräte finden könne (Abb. A.6). Da der Rechner zu diesem Zeitpunkt keinen Brenner angeschlossen hat, ist die Meldung im Prinzip korrekt, aber durch den Hinweis auf den fehlenden SCSI-Bus eher irritierend. Nach dem Einschalten des Geräts und dem Neustart des Rechners funktioniert es dann wie erwartet und der Brenner wird sofort angemeldet (Abb. A.7).

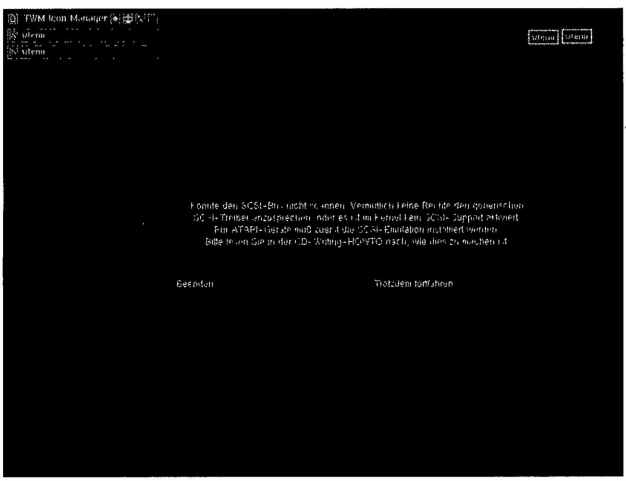


Abb. A.6: Die Fehlermeldung hier auf einem Mac/68k ist nicht korrekt. Der Brenner ist abgeschaltet und kann deshalb nicht gefunden werden, der SCSI-Bus ist natürlich vorhanden

Mit dem Aufruf des Programms und der Erkennung eines CD-Brenners ist es nicht getan. Die verschiedenen Seiten der Einstellungen sind aber sehr gut aufgebaut und werden – soweit möglich – vom Programm auch schon passend ausgefüllt. Ist im Environment die Variable *LANG=de* gesetzt, erschienen die Dialoge und Hinweise auch passend gleich in deutscher Sprache (Abb. A.8).

Es sollen hier nicht alle möglichen GUI-Interfaces für CD-Record ausführlich gewürdigt werden. X-CD-Roast stellt ohnehin einen guten Kompromiß zwischen maximaler Konfiguration und möglichst einfacher Bedienung dar.

Ein weiteres in den Packages enthaltenes Programm ist *gnometoaster*. Es besitzt gegenüber X-CD-Roast deutliche Schwächen. Als erstes fällt unangenehm auf, daß es den ATAPI-Brenner nicht erkennt (Abb. A.11). Man kann das Brenndevice aber manuell eintragen. Bei SCSI-Geräten funktioniert die Erkennung problemlos. Man muß dem Programm – auch wenn es einen Brenner gefunden hat – mitteilen, daß dieser tatsächlich CDs brennen kann. Wer einen SCSI-Brenner verwendet, sollte sich dieses Programm aber durchaus einmal ansehen. Was mich stört, ist die Vielzahl der Optionen (Abb. A.12). Wenn man die Optionen-Dialoge durchblättert, wird man von zahlreichen

Einstellmöglichkeiten beeindruckt. Ob man sie wirklich braucht, darf bezweifelt werden und ob aber noch korrekt gebrannt wird, wenn der unbedarfte Anwender hier an der falschen Schraube dreht?

Abb. A.7: Jetzt ist der Brenner am Macintosh da!

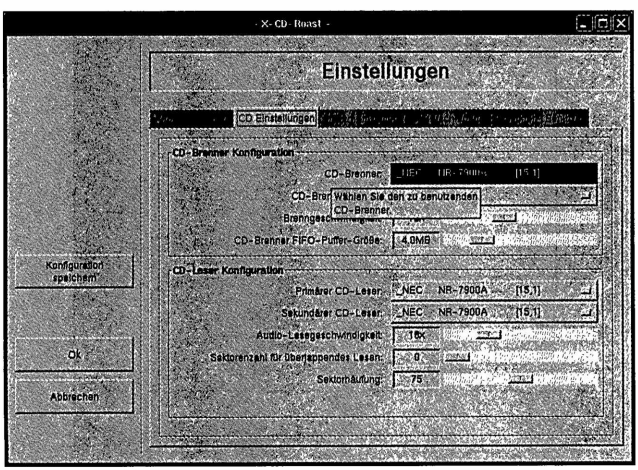


Abb. A.8: Die Einstellungen in X-CD-Roast reichen völlig aus und bieten alle benötigten Optionen

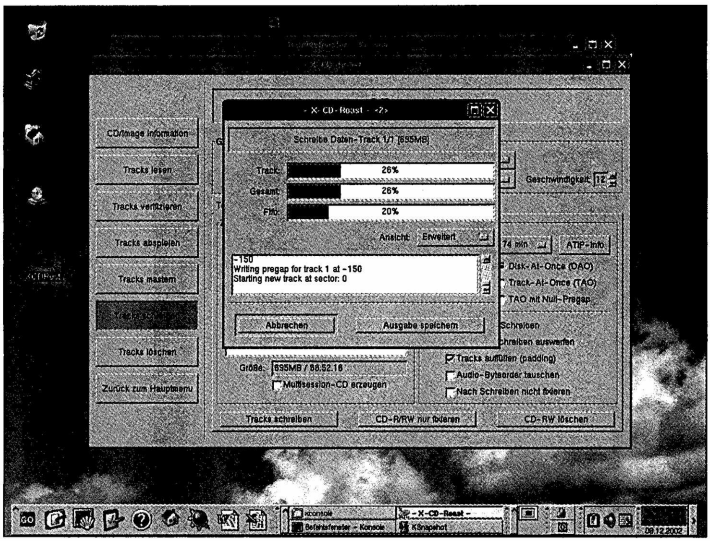


Abb. A.9: Das Brennen einer CD mit X-CD-Roast als Interface zu cdrecord ist sehr komfortabel, auch auf einem ATAPI-Brenner

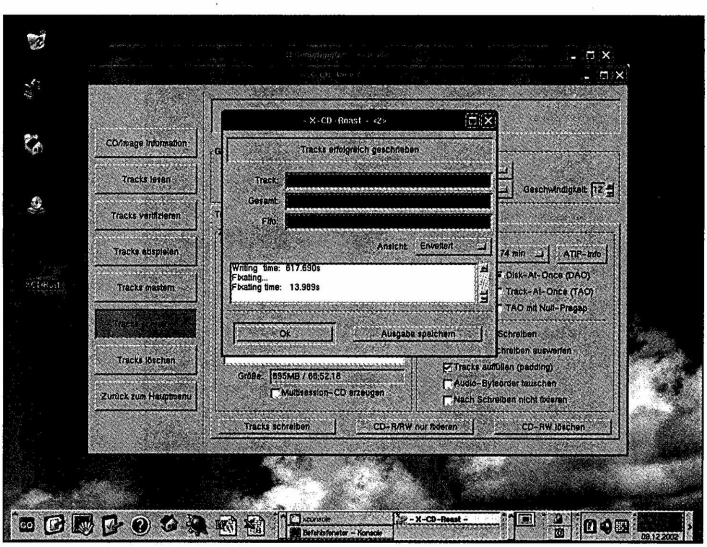


Abb. A.10: Der Brennvorgang wurde erfolgreich abgeschlossen.

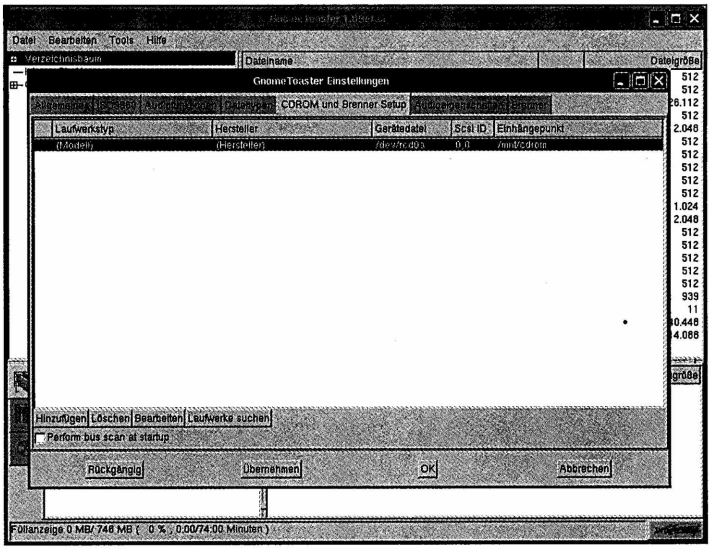


Abb. A.11: Der Gnome-Toaster hat Probleme beim Erkennen von ATAPI-Brennern

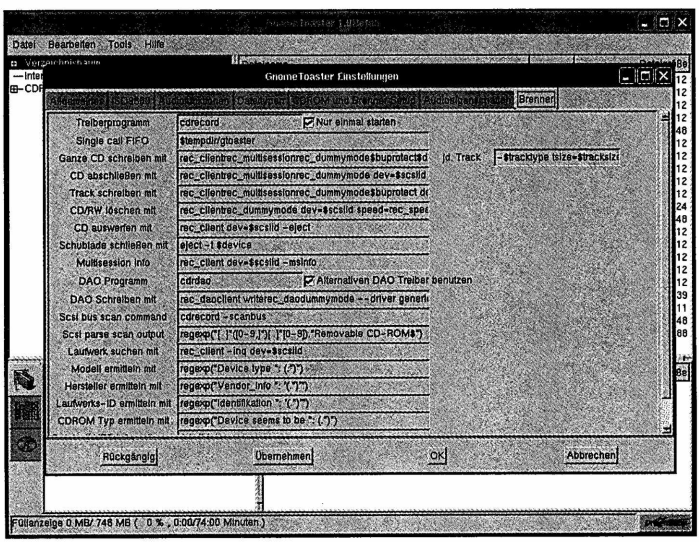


Abb. A.12: Die Optionen beim GnomeToaster sind vielfältig. Hier wäre weniger aber von Vorteil gewesen. Die meisten Anwender werden mit den Optionen kaum etwas anfangen können oder müssen

Als letztes Programm soll hier nur kurz der Gnome CD Master erwähnt werden (Abb. A.13). Damit können keine Daten-CDs gebrannt werden. Als einfaches und schnelles Tool für das Mastern von Audio-CDs ist es aber sehr empfehlenswert.

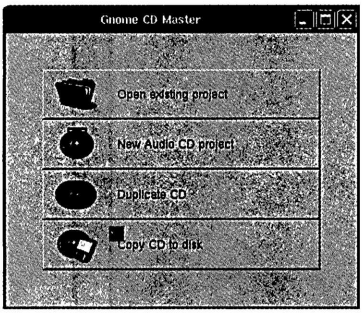


Abb. A.13:
Gnome-CD-Master ist nur für Audio-CDs geeignet und arbeitet auch nur mit SCSI-Geräten. Als Device wird immer die SCSI-ID verlangt, nicht der Pfad zum Gerät

B Systembefehle von NetBSD

Die folgende alphabetische Liste enthält alle Programme, die sich in den Verzeichnissen »/bin«, »/sbin«, »/usr/bin«, und »/usr/sbin« befinden. Die Programme und Daemons in »/sbin« und »/usr/sbin« sind für den Aufruf durch den Systemadministrator reserviert, die Programme in »/bin« und »/usr/bin« können auch von normalen Anwendern (zum Teil mit eingeschränkter Funktionalität gegenüber *root*) verwendet werden.

Befehl	Ort	Funktion/Beschreibung
[	/bin	Test von Bedingungen, siehe auch *test*
ac	/usr/sbin	Verbindungszeit-Zähler
accton	/usr/sbin	System-Account zulassen und abschalten
addftinfo	/usr/bin	Informationen für *groff* zu *troff*-Fontdateien zufügen
addr2line	/usr/bin	Adressen in Dateinamen und Zeilennummern umwandeln
altqd	/usr/sbin	Der AltQ-Daemon
altqstat	/usr/sbin	AltQ-Status anzeigen
amd	/usr/sbin	Der BSD-Automount-Daemon
amq	/usr/sbin	BSD-Automounter-Abfragetool
apm	/usr/sbin	Kontrolle des APM-BIOS und Anzeige des APM-Status
apmd	/usr/sbin	Advanced Power Management Monitor Daemon
apply	/usr/bin	Einen Befehl auf einen Satz von Argumenten ausführen
apropos	/usr/bin	Durchsuchen der *whatis*-Datenbank
ar	/usr/bin	Erzeugen, Ändern und Extrahieren von/aus Archiven
arp	/usr/sbin	Anzeige und Kontrolle der Adreßauflösung
as	/usr/bin	Der GNU-Assembler
asa	/usr/bin	Carriage-Control-Zeichen interpretieren
at	/usr/bin	Befehl für spätere Ausführung zwischenspeichern
atactl	/sbin	Programm zur Bearbeitung von ATA- (IDE-) Laufwerken
atq	/usr/bin	Auflisting der für spätere Ausführung gespeicherten Jobs
atrm	/usr/bin	Löschen zwischengespeicherter Jobs (siehe *at*)
audioctl	/usr/bin	Audio-Device-Kontrollprogramm
audioplay	/usr/bin	Audio-Dateien abspielen
audiorecord	/usr/bin	Audio-Dateien aufnehmen
awk	/usr/bin	Pattern Scanning und Processing Language
bad144	/usr/sbin	Lesen und schreiben der DEC 144-Bad-Sektor-Information

badsect	/sbin	Dateien für schlechte Sektoren erzeugen
banner	/usr/bin	Große Banner auf dem Drucker erzeugen
basename	/usr/bin	Rückgabe des reinen Dateinamens aus einem Pfadnamen
batch	/usr/bin	Ausführen von für spätere Ausführung vermerkten Jobs
bc	/usr/bin	Ein Rechner
bdes	/usr/bin	Ver- und Entschlüsseln mit DES-Algorithmen
biff	/usr/bin	Benachrichtigung bei Mail-Empfang ein- und ausschalten
bootpd	/usr/sbin	Der Internet Boot-Protokoll-Server (siehe *bootpgw*)
bootpef	/usr/sbin	BOOTP-Erweiterungsdateien-Compiler
bootpgw	/usr/sbin	Der Internet Boot-Protokoll-Gateway (siehe *bootpd*)
bootptest	/usr/sbin	Senden von BOOTP-Anfragen und Drucken von Anworten
brconfig	/sbin	Netzwerk-Bridge-Parameter konfigurieren
bunzip2	/usr/bin	Entpacker für bz2-Archive
bzcat	/usr/bin	Entpacker für bz2-Archive nach »stdout«
bzip2	/usr/bin	Bzip2-Packer
bzip2recover	/usr/bin	Restauriert Daten aus defekten Bzip2-Archiven
c++	/usr/bin	GNU Project C und C++ Compiler, siehe auch *cc*, *CC*, *gcc* und *g++*
c++filt	/usr/bin	C++-Symbol-Demangler
c89	/usr/bin	Posix.2-C-Compiler
cal	/usr/bin	Einfache Kalender-Anzeige (siehe auch *ncal*)
calendar	/usr/bin	Terminerinnerung
cap_mkdb	/usr/bin	Erzeugen der CAP-Datenbank
cat	/bin	Abschneiden und Anzeigen von Dateien
catman	/usr/bin	Vorformatieren von Manpages
catman	/usr/sbin	Formatieren von Cat-Seiten aus Manpages
cc	/usr/bin	GNU Project C und C++ Compiler, siehe auch *c++*, *CC*, *g++* und *gcc*
CC	/usr/bin	GNU Project C und C++ Compiler, siehe auch *c++*, *cc*, *g++* und *gcc*
ccdconfig	/sbin	Konfigurationsprogramm des Treibers für verbundene Datenträger
cdplay	/usr/bin	Abspielprogramm für Audio-CDs
chat	/usr/sbin	Programm zur Kommunikation zwischen Rechner und Modem

checknr	/usr/bin	Prüfen von *nroff/troff*-Dateien
chflags	/usr/bin	Ändern von Dateiflags
chfn	/usr/bin	Siehe *chpass*
chgrp	/usr/bin	Ändern der Gruppenzugehörigkeit von Dateien
chio	/bin	Hilfsprogramm für die Kontrolle von Wechselmedien
chmod	/bin	Ändern der Datei- und Verzeichnisattribute
chown	/usr/sbin	Ändern der Userzugehörigkeit einer Datei oder eines Verzeichnisses (Change Owner)
chpass	/usr/bin	Ändern der Datenbankeinträge für einen User (auch als *chfn*, *chsh*, *ypchpass*, *ypchfn* und *ypchsh* gelinkt)
chroot	/usr/sbin	Ändern des Rootverzeichnisses für einen Prozeß
chrtbl	/usr/sbin	Erzeugen der Zeichenklassifikation und von Groß-/Kleinbuchstaben-Umwandlungstabellen
chsh	/usr/bin	Shell Ändern. Synonym für *chpass*
ci	/usr/bin	RCS-Revisionen einchecken
cksum	/usr/bin	Anzeigen von Checksummen und Zählen von Blöcken
clear	/usr/bin	Löschen des Terminal-Bildschirms
clri	/sbin	Löschen eines Inodes
cmp	/usr/bin	Vergleichen von zwei Dateien
cnwctl	/usr/sbin	Anzeige der Statistik und Kontrolle von Netwave AirSurfer PC-Karten
co	/usr/bin	RCS-Revisionen auschecken
col	/usr/bin	Filtern von Zeilenumbrüchen aus der Eingabe
colcrt	/usr/bin	Filtern von nroff-Ausgaben für die Voranzeige auf dem Bildschirm
colrm	/usr/bin	Spalten aus einer Datei löschen
column	/usr/bin	Formatieren von Eingaben in mehrere Spalten
comm	/usr/bin	Auswählen oder Ablehnen von Zeilen, die in zwei Dateien gleichzeitig enthalten sind
compile_et	/usr/bin	Fehlertabellencompiler
compress	/usr/bin	Unix-Komprimierprogramm (Endung ».Z«)
config	/usr/sbin	Kernel-Konfigurationsprogramm
cp	/bin	Kopieren von Dateien und/oder Verzeichnissen
cpio	/usr/bin	Dateien in und aus Archiven kopieren
cpp	/usr/bin	Der GNU-C-kompatible Präprozessor

cron	/usr/sbin	Daemon für das Ausführen zeitdefinierter Programme
crontab	/usr/bin	Verwalten von *crontab*-Dateien für individuelle User
crunchgen	/usr/bin	Erzeugen der Umgebung für ein zusammengefaßtes Binary, das mehrfache Befehle enthält
crunchide	/usr/bin	Versteckt symbolische Namen vor dem *ld*, benötigt für zusammengefaßte Programme
csh	/bin	Die C-Shell
ctags	/usr/bin	Erzeugen einer Tag-Datei
cu	/usr/bin	Aufrufen eines anderen Systems und als Einwahlterminal agieren
cut	/usr/bin	Teile jeder Zeile einer Datei tabellarisch auswählen
cvs	/usr/bin	Versionskontrollsystem
cvsbug	/usr/bin	Fehlerreports via CVS an eine zentrale Support-Site senden
daicctl	/usr/sbin	ISDN-Kontrolle, Test- und Statistik-Programm
date	/bin	Anzeige des Datums und der Uhrzeit
dbsym	/usr/sbin	Kopieren der Symboltabelle des Kerneln in den Bereich »db_symtab«
dc	/usr/bin	Ein UPN-Rechner
dd	/bin	Kopier- und Imagingprogramm
dev_mkdb	/usr/sbin	Erzeugen der Datenbank der Geräte in »/dev«
df	/bin	Anzeige des freien Speichers gemounteter Datenträger
dhclient	/sbin	Client für das DHCP-Protokoll
dhclient-script	/sbin	Skript für die Konfiguration des DHCP-Clients
dhcpd	/usr/sbin	Der DHCP-Server
dhcrelay	/usr/sbin	Relay-Agent für DHCP
diff	/usr/bin	Unterschiede zwischen zwei Dateien ermitteln
diff3	/usr/bin	Unterschiede zwischen drei Dateien ermitteln
dig	/usr/bin	Senden von Domainnamen-Abfragepaketen an Name-Server
dirname	/usr/bin	Liefern des Pfadanteils eines Pfadnamens (siehe *basename*)
disklabel	/sbin	Der Standard-BSD-Disklabel-Editor
diskpart	/usr/sbin	Kalkulieren der voreingestellten Plattenpartitionsgrößen
dkctl	/sbin	Datenträgerkonfigurationsprogramm
dmesg	/sbin	Anzeige des Meldungspuffers des Systems
dnsquery	/usr/bin	Abfrage von Domain Name Servern über BIND-Resolver-Aufrufe

domainname	/bin	Abfrage und Setzen des Domainnamens
doscmd	/usr/bin	Verwenden eines Subsets von Real-Mode-DOS-Befehlen
dtmfdecode	/usr/sbin	Teil des ISDN4BSD-Pakets: Erkennen von DTMF-Tönen im Audio-Stream
du	/usr/bin	Anzeige des Speicherverbrauchs von Dateien und Verzeichnissen
dump	/sbin	Dateisystem-Backup-Programm (siehe auch *rdump*)
dump_lfs	/sbin	Backup-Programm für LFS-Dateisysteme
dumpfs	/sbin	Auslesen der Dateisystem-Informationen
dumpfs	/usr/sbin	Dump der Dateisystem-Informationen
dumplfs	/usr/sbin	Dump der LFS-Dateisystem-Informationen
echo	/bin	Schreibe die Programmparameter auf die Standardausgabe
ed	/bin	Zeilenorientierter Texteditor
edquota	/usr/sbin	Editieren von User-Quotas
egrep	/usr/bin	Entspricht *grep -E*, siehe *grep*
eject	/usr/bin	Wirft einen Wechseldatenträger (Floppy, CD oder Band) aus
env	/usr/bin	Ausgeben und Setzen der Umgebungsvariablen
envstat	/usr/sbin	Anzeige der Werte eines Geräts, das das ENVSYS-API unterstützt
eqn	/usr/bin	Formatieren von Gleichungen für *troff*
error	/usr/bin	Analyseprogramm für Compilerfehlermeldungen
eshconfig	/usr/sbin	Konfiguration des Essential Communications HIPPI-Netzwerk-Interfaces
etcupdate	/usr/sbin	Update der Konfigurationsdateien und Startskripten in »/etc«
ex	/usr/bin	Texteditor (zeilenorientierte Variante von *vi*)
expand	/usr/bin	Expandiere Leerzeichen in Tabulatoren (siehe auch *unexpand*)
expr	/bin	Auswerten von Ausdrücken
f77	/usr/bin	Der GNU Fortan-Compiler. Siehe auch *g77 und fort77*
faithd	/usr/sbin	Der FAITH-IPv6/v4-Übersetzer-Daemon
false	/usr/bin	Rückgabe des Werts »false«
fastboot	/sbin	Alias für den Befehl *reboot*
fasthalt	/sbin	Alias für den Befehl *halt*
fdformat	/usr/bin	Diskettenformatierprogramm
fdisk	/sbin	Festplatten-Verwaltungsprogramm (Anlegen und Löschen von BSD-Slices)

fgen	/usr/bin	IEEE 1275 Open Firmware Fcode Tokenizer
fgrep	/usr/bin	Entspricht *grep -F*, siehe *grep*
file	/usr/bin	Dateityp ermitteln
find	/usr/bin	Suchen von Dateien
finger	/usr/bin	Programm zum Ermitteln von Userinformationen
fixmount	/usr/sbin	Korrigieren von Remote-Mount-Einträgen
flex	/usr/bin	Schneller lexikalischer Analysegenerator
flex++	/usr/bin	Siehe *flex*
fmt	/usr/bin	Einfacher Text-Formatierer
fold	/usr/bin	Einfügen von Zeilenumbrüchen in eine Datei
fort77	/usr/bin	Der GNU Fortan-Compiler. Siehe auch *g77 und f77*
fpr	/usr/bin	Druckprogramm für Fortran-Quelltexte
from	/usr/bin	Ausgabe des Namens eines Mail-Senders
fsck	/sbin	Prüfung der Dateisystem-Konsistenz und interaktive Reparatur
fsck_ext2fs	/sbin	Prüfung der Dateisystem-Konsistenz und interaktive Reparatur für Linux-Ext2-Dateisysteme. Über das Packagesystem sind außerdem die Programme *fsck.ext2* und *fsck.ext3* installierbar (e2fsprogs). Sie befinden sich dann in »/usr/pkg/sbin«.
fsck_ffs	/sbin	Prüfung der Dateisystem-Konsistenz und interaktive Reparatur für BSD-FFS-Dateisysteme
fsck_lfs	/sbin	Prüfung der Dateisystem-Konsistenz und interaktive Reparatur für NetBSD-LFS-Dateisysteme
fsck_msdos	/sbin	Prüfung der Dateisystem-Konsistenz und interaktive Reparatur für das MS-DOS-/Windows-FAT-Dateisysteme
fsdb	/sbin	Debugging- und Editier-Tool für das FFS-Dateisystem
fsinfo	/usr/sbin	Informationsprogramm für den X-Fontserver
fsirand	/sbin	Erhöhung der Sicherheit des Dateisystems durch die Vergabe zufälliger Inode-Nummern
fsplit	/usr/bin	Splitter für Multi-Routinen-Fortran-Dateien in einzelne Dateien
fstat	/usr/bin	Dateistatus anzeigen
ftp	/usr/bin	ARPANET-Dateitransfer-Programm (FTP-Client)
g++	/usr/bin	Der GNU Project C and C++ Compiler, siehe auch *c++*, *cc*, *CC* und *gcc*
g77	/usr/bin	Der GNU Fortan-Compiler. Siehe auch *fort77 und f77*
gcc	/usr/bin	Der GNU Project C und C++ Compiler, siehe auch *c++*, *cc*, *CC* und *g++*

gcov	/usr/bin	Hilfsprogramm für den GNU C++-Compiler
gcpp	/usr/bin	GNU CPP-Präprozessor (siehe auch *cpp* und *cccp*)
gdb	/usr/bin	Der GNU-Debugger
gencat	/usr/bin	NLS-Katalog-Compiler
getconf	/usr/bin	Lesen von Konfigurationswerten
getencstat	/usr/sbin	SCSI-Umgebungsservice-Status abfragen
getopt	/usr/bin	Parser für Kommandooptionen
gettext	/usr/bin	Das GNU-Hilfsprogramm gettext
gprof	/usr/bin	Generieren von Profiler-Daten von C-, Fortran- und Pascal-Programmen
greconfig	/usr/sbin	Werte für das Pseudo-Device *gre* setzen
grep	/usr/bin	Ausgabe von Zeilen, die einen Ausdruck enthalten. Siehe auch *egrep*, *fgrep* und *zgrep*
grn	/usr/bin	groff-Präprozessor für Gremlin-Dateien
grodvi	/usr/bin	Konvertieren von groff-Output in das TeX-DVI-Format
groff	/usr/bin	Frontend für das *groff*-Dokumenten-Formatier-System
grog	/usr/bin	Auswahlprogramm für Optionen des *groff*-Befehls
grohtml	/usr/bin	HTML-Treiber für *groff*
grolbp	/usr/bin	*groff*-Treiber für CanonCAPSL-Drucker
grolj4	/usr/bin	*groff*-Treiber für HP-Laserjet 4
grops	/usr/bin	Postscript-Treiber für *groff*
grotty	/usr/bin	*groff*-Treiber für schreibmaschinenartige Geräte
group	/usr/sbin	Gruppenverwaltungsprogramm
groupadd	/usr/sbin	Gruppe in das System einfügen
groupdel	/usr/sbin	Gruppe vom System löschen
groupinfo	/usr/sbin	Gruppeninformationen anzeigen
groupmod	/usr/sbin	Vorhandene Gruppe ändern
groups	/usr/bin	Gruppenmitgliedschaft eines Users anzeigen
gspa	/usr/sbin	GCC-Präprozessor
gspahextoc	/usr/sbin	GCC-Präprozessor
gunzip	/usr/bin	Entpackprogramm für ».gz«-Dateien
gzcat	/usr/bin	Entspricht einem Aufruf von *gunzip* -c
gzexe	/usr/bin	Erzeugen gepackter ausführbarer Programme
gzip	/usr/bin	Packer für ».gz«-Dateien

halt	/sbin	Anhalten des Systems. Siehe auch *fasthalt* und *shutdown*
head	/usr/bin	Anzeigen der ersten Zeilen einer Datei
hesinfo	/usr/bin	Hesiod-Datenbank analysieren
hexdump	/usr/bin	Hexdump-Utility (siehe auch *hd*)
hlfsd	/usr/sbin	Der Home-Link-Dateisystem-Daemon
host	/usr/bin	Suche nach Hosts unter Verwendung des Domain Servers
hostname	/bin	Anzeige und Setzen des Hostnamens
hoststat	/usr/bin	Link auf */usr/sbin/mailwrapper*
hpftodit	/usr/bin	Erzeugen von Fontbeschreibungen für *groff*
hprop	/usr/sbin	Daten in Heimdal-Datenbank eintragen
id	/usr/bin	Ausgeben der User- und Gruppen-ID eines anzugebenden Users
ident	/usr/bin	Erkennung von RCS-Schlüsselworten in Dateien
ifconfig	/sbin	Konfiguration der Parameter der Netzwerk-Interfaces
ifmcstat	/usr/sbin	Dump der Multicast-Gruppenmanagement-Statistik pro Interface
ifwatchd	/usr/sbin	Überwachen von Adressen, die zu Interfaces hinzugefügt oder von ihnen entfernt wurden und Aufrufen der entsprechenden Start- und Stop-Skripten
indent	/usr/bin	Formatierprogramm für C-Quelltexte
indxbib	/usr/bin	Erzeugen eines invertierten Indexes für bibliographische Datenbanken
inetd	/usr/sbin	Der Internet Super Server
info	/usr/bin	Leseprogramm für Info-Dokumente
init	/sbin	Initialisierungs- und Modus-Umschalt-Programm nach Unix-System-V. Bei NetBSD wegen des Simple-Init-Systems mit deutlich eingeschränkter Funktionalität
innetgr	/usr/bin	Netgroup-Mitglieder prüfen
install	/usr/bin	Installationsprogramm für Binärdateien
installboot	/usr/sbin	Installation des Bootstrapper-Programms
install-info	/usr/bin	Aktualisierung von INFO/DIR-Einträgen
install-sid	/usr/bin	Installationsprogramm für Binärdateien
iopctl	/usr/sbin	Programm für die Kontrolle der IOP-Devices
iostat	/usr/sbin	Anzeigen der Ein-/Ausgabe-Statistik
ipcrm	/usr/bin	Entfernen von Message Queues, Semaphoren und Shared-Memory-Segmenten

ipcs	/usr/bin	Ausgeben der Daten der System-V-Interprozeß-Kommunikation
ipf	/sbin	Ändern der Packet-Filter-Listen für IP-Packet-Input und -Output
ipfs	/usr/sbin	Sichern und Wiederherstellen der Informationen für NAT und die Statustabellen
ipfstat	/sbin	Ausgabe der Packet-Filter-Statistik und der Filterliste
ipfstat	/usr/sbin	Reporte zur Packet-Filtering-Statistik und der Filterlisten
ipftest	/usr/sbin	Testen der Packet-Filter-Regeln
ipfw	/sbin	User-Interface für die Kontrolle von *ipfirewall* und *dummynet*
ipmon	/usr/sbin	Überwachen von »/dev/ispl« auf geloggte Pakete
ipnat	/usr/sbin	Benutzerinterface für den NAT
ipppctl	/sbin	Anzeigen und Setzen der Parameter für das ISDN-Interface
ipresend	/usr/sbin	Sendet IP-Packets aus dem Netzwerk zurück
ipsend	/usr/sbin	Sendet IP-Packets
iptest	/usr/sbin	Automatisches Generieren eines Packets zur Prüfung der IP-Funktionalität
irdaattach	/usr/sbin	Serielle Verbindungen an den IrDA-Treiber binden
isdnd	/usr/sbin	ISDN-Verbindungs-Daemon
isdnmonitor	/usr/sbin	*isdnd*-Remote-Monitor
isdntel	/usr/sbin	ISDN-Telefon-Anrufbeantworter
isdntelctl	/usr/sbin	ISDN-Telefon-Soundformatkonvertierung
isdntrace	/usr/sbin	ISDN-Protokollverfolgungstool
join	/usr/bin	Relationaler Datenbank-Operator
jot	/usr/bin	Gebe sequentielle oder zufällige Daten aus
kadmin	/usr/sbin	Kerberos-Administrationstool
kdc	/usr/sbin	Kerberos-V-Server
kdestroy	/usr/bin	Zerstören des aktuellen Kerberos-Ticket-Files
kdump	/usr/bin	Anzeigen der Trace-Daten des Kernels
kf	/usr/bin	Ticket sicher weiterleiten
kgmon	/usr/sbin	Erzeuge eines Dumps der Profile-Puffer des Systems
kill	/bin	Beenden eines Prozesses
kinit	/usr/bin	Anfordern des grundlegenden Kerberos-Tickets
klist	/usr/bin	Listen der aktuellen Kerberos-Tickets in der Ticket-Datei

kpasswd	/usr/bin	Paßwort-Programm für Kerberos V
ksh	/bin	Public Domain Korn Shell
kstash	/usr/sbin	Sichern des Kerberos-4-Master-Paßworts in einer Datei
ktrace	/usr/bin	Einschalten der Kernel-Prozeßverfolgung
ktruss	/usr/bin	Kernel-Prozeß-Tracing einschalten
ktutil	/usr/sbin	Verwaltung von Kerberos-Schlüsseltabellen
kvm_mkdb	/usr/sbin	Kernel-Datenbank erzeugen
lam	/usr/bin	Laminieren von Dateien
last	/usr/bin	Ermitteln des letzten Logins eines Users und der ttys
lastcomm	/usr/bin	Anzeigen der letzten Befehlsaufrufe in umgekehrter Reihenfolge. Benötigt die Account-Datei »/var/account/acct«
lastlogin	/usr/sbin	Ermitteln des letzten Loginzeitpunkts von Usern
ld	/usr/bin	Der GNU-Linker
ldconfig	/sbin	Konfigurationsprogramm für den Cache der Shared Libraries
ldd	/usr/bin	Auflisten der Abhängigkeiten dynamischer Objekte
leave	/usr/bin	Erinnert an den Zeitpunkt, zu gehen
less	/usr/bin	Das Gegenteil von more
lessecho	/usr/bin	Variante des Befehls *echo*
lesskey	/usr/bin	Programm für die Tastenzuweisung von *less*
lex	/usr/bin	Lexikalischer Analysegenerator (siehe auch *flex*)
link	/bin	Erstellen von Links (siehe auch *ln*)
link	/usr/sbin	Aufruf der Funktion *link()*
lint	/usr/bin	Ein Testprogramm für die Richtigkeit von C-Quelltexten
lkbib	/usr/bin	Durchsuchen bibliographischer Datenbanken
lmcctl	/sbin	Konfigurationsprogramm für Lan Media SSI-, HSSI- und DS3-Karten
ln	/bin	Erstellen von Links (siehe auch *link*)
locate	/usr/bin	Schnelle Suche von Dateinamen anhand der *locate*-Datenbank. Diese muß mit */usr/libexec/locate.updatedb* aufgebaut werden
lock	/usr/bin	Reservieren eines Terminals
logger	/usr/bin	Einfügen von Einträgen in das System-Log
login	/usr/bin	Einloggen in den Computer
logname	/usr/bin	Ausgeben des Namens des Users beim Login. Siehe auch *who* und *whoami*

look	/usr/bin	Anzeigen von Zeilen, die mit einem angegebenen String beginnen
lookbib	/usr/bin	Suchen in einer bibliographischen Dankenbank
lorder	/usr/bin	Listen der Abhängigkeiten von Objektdateien
lp	/usr/bin	Frontend für den Druckerspooler
lpc	/usr/sbin	Druckerkontrollprogramm
lpd	/usr/sbin	Druckerspooler-Daemon
lpq	/usr/bin	Spooler-Queue-Überwachungsprogramm
lpr	/usr/bin	Offline-Drucker
lptest	/usr/sbin	Erzeugen von Druckertestdaten. Kann auch für die Prüfung von Terminals verwendet werden
ls	/bin	Auflisten von Verzeichnisinhalten
m4	/usr/bin	Makrosprachen-Prozessor
Mail	/usr/bin	siehe »mail«
mail	/usr/bin	Senden und Empfangen von Mails, auch unter den Namen »Mail« und »mailx« verfügbar
mailq	/usr/bin	Ausgeben der Mail-Warteschlange (Link auf »/usr/sbin/mail-wrapper«)
mailstats	/usr/sbin	Anzeigen der Mail-Statistik
mailwrapper	/usr/sbin	Aufrufen des entsprechenden Mail Transfer Agents, basierend auf der Konfigurationsdatei
mailx	/usr/bin	Senden und Empfangen von Mails (siehe *mail*)
make	/usr/bin	Verwaltung von Programmen und Programmquelltexten
makedbm	/usr/sbin	Erzeuge eine YP-Datenbank
makefs	/usr/sbin	Erzeugen eines Dateisystemimages aus einem Verzeichnisbaum
makeinfo	/usr/bin	Übersetzen von Texinfo-Dokumenten
makemap	/usr/sbin	Erzeugen von Datenbank-Maps für Sendmail
man	/usr/bin	Das Anzeigeprogramm für Manual-Seiten
map-mbone	/usr/sbin	Multicast-Verbindungs-Mapper
mbrlabel	/sbin	Aktualisierung des Disklabels aus dem Master Boot Record
md2	/usr/bin	Checksummen anzeigen(siehe *cksum*)
md4	/usr/bin	Checksummen anzeigen(siehe *cksum*)
md5	/usr/bin	Checksummen anzeigen(siehe *cksum*)
mdconfig	/usr/sbin	Konfigurationswerkzeug für RAM-Disks

mdsetimage	/usr/sbin	Setzen eines Kernel-RAM-Disk-Images
menuc	/usr/bin	Menücompiler für curses-basierte Menüs
merge	/usr/bin	Dreiwege-Dateizusammenführung
mesg	/usr/bin	Anzeigen und Nicht-Anzeigen von Meldungen anderer Anwender
midiplay	/usr/bin	MIDI-Dateien abspielen
mixerctl	/usr/bin	Kontrollprogramm für den Audio-Mixer
mk_cmds	/usr/bin	Erzeugen von Befehlen aus einer Tabellendatei
mkalias	/usr/sbin	Konvertiert »mail.alias-Maps« in »mail.byaddr«-Maps
mk-amd-map	/usr/sbin	Erzeugen von Datenbank-Maps für *amd*
mkdep	/usr/bin	Erzeugen einer Abhängigkeitenliste für Make-Dateien
mkdir	/bin	Erzeugen eines Verzeichnisses
mkfifo	/usr/bin	Erzeugen von FIFOs
mklocale	/usr/bin	Erzeugen von *LC_CTYPE*-Locale-Dateien
mknetid	/usr/sbin	Ein YP-Filter-Programm. Erzeugt eine Map mit dem Namen »netid.byname«
mknod	/sbin	Erzeugen einer Spezialdatei in »/dev«
mkstr	/usr/bin	Erzeugen einer Fehlermeldungsdatei aus C-Quellen
mktemp	/usr/bin	Erzeugen einer einzigartigen Temporärdatei
mld6query	/usr/sbin	Erzeugen einer Multicast-Lauscher-Anfrage
mlxcontrol	/usr/sbin	Verwaltungstool für Mylex-RAID-Controller der DAC-Familie
modload	/sbin	Kernel-Modul-Ladeprogramm
modstat	/usr/bin	Statusanzeige für geladene Kernel-Module
modunload	/sbin	Kernel-Modul-Entladeprogramm
mopchk	/usr/sbin	MOP-Check-Hilfsprogramm
mopd	/usr/sbin	MOP-Lader-Daemon
mopprobe	/usr/sbin	MOP-Testprogramm
moptrace	/usr/sbin	MOP-Trace-Utility
more	/usr/bin	Seitenweises Anzeigen von Textdateien
mount	/sbin	Einbinden (mounten) von Datenträgern in den Verzeichnisbaum (siehe auch *umount*)
mount_cd9660	/sbin	Einbinden (mounten) von CD-ROMs
mount_ados	/sbin	Einbinden (mounten) eines AmigaDOS-Dateisystems

mount_ext2fs	/sbin	Einbinden (mounten) von Datenträgern im Linux-Ext2-Dateisystem
mount_fdesc	/sbin	Einbinden (mounten) des fdesc-Dateisystems
mount_ffs	/sbin	Einbinden (mounten) des Standard-Dateisystems FFS von NetBSD
mount_file-core	/sbin	Einbinden (mounten) des Filecore-Dateisystems
mount_kernfs	/sbin	Einbinden (mounten) des Kernel-Dateisystems
mount_lfs	/sbin	Einbinden (mounten) des Log Structured Dateisystems
mount_mfs	/sbin	Erzeugen eines Dateisystems im virtuellen Speicher (RAM-Disk)
mount_msdos	/sbin	Einbinden eines FAT- oder VFAT-Dateisystems von DOS/Windows
mount_nfs	/sbin	Einbinden von fernen Verzeichnisfreigaben (NFS)
mount_ntfs	/sbin	Einbinden (mounten) von NTFS-Partitionen (Windows NT, Windows 2000). Die Partitionen sollten nur read-only gemountet werden. Die Unterstützung des Dateisystems ist nicht vollständig ausgereift
mount_null	/sbin	Einbinden eines Loopback-Dateisystem-Unterbaums
mount_overlay	/sbin	Einbinden (mounten) eines Overlay-Dateisystems (ähnlich wie Union)
mount_portal	/sbin	Einbinden (mounten) eines Portal-Dateisystems
mount_procfs	/sbin	Einbinden (mounten) des FreeBSD-Prozeßdateisystems
mount_ufs	/sbin	Einbinden (mounten) des Standard-Dateisystems FFS (vormals UFS) von NetBSD
mount_umap	/sbin	Beispiel für eine Dateisystem-Layer. Damit wird ein Unterbaum eingemountet
mount_union	/sbin	Überlagerung von Dateisystemteilen auf andere
mountd	/usr/sbin	NFS-Mount-Daemon
moused	/usr/sbin	Der Maus-Daemon der Textkonsole
mrinfo	/usr/sbin	Anzeige der Konfigurationsinformationen von einem Multicast-Router
mrouted	/usr/sbin	Der IP-Multicast-Routing-Daemon
mset	/usr/bin	Ermitteln des ASCII-nach-IBM 3270 Tastaturmappings
msgc	/usr/bin	Compiler für das curses-basierte Anzeigesystem
msgcmp	/usr/bin	Vergleichsprogramm für zwei ».po-Dateien«
msgcomm	/usr/bin	Suche Meldungen in zwei oder mehr ».po«-Dateien

msgfmt	/usr/bin	Erzeuge binäre Dateien mit Meldungen aus der textuellen Beschreibung
msgmerge	/usr/bin	Verschmelze zwei ».po«-Dateien im Uniforum-Format
msgs	/usr/bin	System-Meldungen und Junk-Mail-Programm
msgunfmt	/usr/bin	Wandle binäre ».mo«-Dateien zurück in ».po«-Dateien im Uniforum-Format
mt	/bin	Bandlaufwerke bearbeiten
mtree	/usr/sbin	Anzeigen einer Verzeichnishierarchie
mv	/bin	Umbenennen und Verschieben von Dateien und Verzeichnissen
named	/usr/sbin	Der Internet-Domain-Name Server (DNS)
named.reload	/usr/sbin	Weist den Name Server an, seine Datenbank zu synchronisieren
named.restart	/usr/sbin	Anhalten und Neustart des Name Servers
named-bootconf	/usr/sbin	Konvertierprogramm für Name-Server-Konfigurationsdateien
ndbootd	/usr/sbin	Protokollserver für Sun Netzwerk-Disks
ndc	/usr/sbin	Name-Daemon-Kontrollprogramm
ndp	/usr/sbin	Kontrolle und Diagnose für das IPv6-Nachbarschafts-Untersuchungs-Protokoll (NDP)
neqn	/usr/bin	Formatieren von Gleichungen für die ASCII-Ausgabe. Siehe *eqn*
netgroup_mkdb	/usr/sbin	Erzeugt eine Netgroup-Datenbank
netstat	/usr/bin	Anzeige des Netzwerk-Status
newaliases	/usr/bin	Neuerzeugen der Datenbank für die Mail-Aliases-Datei (Link auf */usr/sbin/mailwrapper*)
newbtconf	/sbin	Multiboot-Konfigurationstool
newfs	/sbin	Konstruieren (formatieren) eines neuen FFS-Dateisystems
newfs_lfs	/sbin	Formatierprogramm für das LFS-Dateisystem
newfs_msdos	/sbin	Formatieren einer FAT-Partition
newsyslog	/usr/bin	System-Logdateien in verwaltbare Größen bringen
nfsd	/usr/sbin	Remote NFS-Server
nfsstat	/usr/bin	Anzeigen der NFS-Statistik
nice	/usr/bin	Ausführen eines Programm mit niedriger Priorität
nl	/usr/bin	Zeilennummernfilter

nm	/usr/bin	Symbole aus einer Objektdatei auflisten
nohup	/usr/bin	Aufruf von Befehlen, die immun gegen Aufhänger sind
nologin	/sbin	Anzeige einer Meldung an User, die sich nicht einloggen dürfen
nroff	/usr/bin	Emulation des *nroff*-Befehls mit *groff*
nslookup	/usr/bin	Interaktive Suche von Internet Name Servern
nsupdate	/usr/bin	Interaktive Aktualisierung von Internet Name Servern
ntpd	/usr/sbin	Daemon für das Network Time Protocol (NTP)
ntpdate	/usr/sbin	Setzen des Datums und der Uhr via NTP
ntpdc	/usr/sbin	Besonderes NTP-Abfrage-Programm
ntpq	/usr/sbin	Standard NTP-Abfrageprogramm
ntptime	/usr/sbin	Auslesen der Kernel-Zeitvariablen
ntptimeset	/usr/sbin	Programm zum Setzen der Zeit via Network Time Protocol
ntptrace	/usr/sbin	Verfolgen einer Kette von NTP-Servern zurück auf die ursprüngliche Quelle
objcopy	/usr/bin	Kopieren und Übersetzen von Objektdateien
objdump	/usr/bin	Anzeige von Informationen über Objektdateien
od	/usr/bin	Link auf das Programm hexdump, das das ältere Programm *od* ersetzt
omshell	/usr/sbin	OMAPI-Kommandoshell
openssl	/usr/bin	OpenSSL-Kommandozeilen-Tool
pac	/usr/sbin	Drucker- und Plotter-Informationen
page	/usr/bin	Synonym für *less* und das Gegenteil von *more*
pagesize	/usr/bin	Ausgabe der Seitengröße des Systems
passwd	/usr/bin	Definition und Änderungen von Benutzerpaßworten
paste	/usr/bin	Zusammenführen korrespondierender Zeilen von Dateien
patch	/usr/bin	Ausführen einer *diff*-Datei auf eine Originaldatei
pawd	/usr/bin	Ausgabe des Automounter-Arbeitsverzeichnisses
pax	/bin	Lesen und Schreiben von Archiven und Kopieren von Verzeichnishierarchien
pcictl	/usr/sbin	Programm zur Manipulation des PCI-Busses
pfbtops	/usr/bin	Übersetzer für einen Postscript-Font im ».pfb«-Format nach ASCII
pgrep	/usr/bin	Finden von Prozessen anhand des Namens (siehe *pkill*)
pic	/usr/bin	Kompiliere Bilder für *troff* oder TeX

ping	/sbin	Sende ICMP-Anfragen an einen Netzwerk-Host
ping6	/sbin	Sende ICMPv6-Anfragen an einen Netzwerk-Host
pkg_add	/usr/sbin	Einbinden binärer Pakete in das NetBSD-System
pkg_admin	/usr/sbin	Administrationswerkzeug für das Paketsystem
pkg_create	/usr/sbin	Erzeugen von Software-Paketen für die Binärdistribution
pkg_delete	/usr/sbin	Löschen von installierten Softwarepaketen aus dem System
pkg_info	/usr/sbin	Anzeige von Informationen zu Softwarepaketen
pkill	/usr/bin	Entfernen von Prozessen anhand des Namens (siehe *pgrep*)
pmc	/usr/bin	Performance-Monitor für Prozesse, ähnlich dem Befehl *time*
postalias	/usr/sbin	Datenbankverwaltung von Postfix-Aliasen
postcat	/usr/sbin	Anzeigen des Inhalts des Dateiinhalts der Postfix-Queue
postconf	/usr/sbin	Konfigurationsprogramm für Postfix
postdrop	/usr/sbin	Mail-Posting-Utilility von Postfix
postfix	/usr/sbin	Postfix-Kontrollprogramm
postkick	/usr/sbin	Postfix-Service anstoßen
postlock	/usr/sbin	Sperren des Postfix-Mailordners und Befehl ausführen
postmap	/usr/sbin	Tool für das Postfix-Abfrage-Management
postqueue	/usr/sbin	Postfix-Queue-Kontrolle
postsuper	/usr/sbin	Postfix-Superintendent
poweroff	/sbin	Shutdown-Programm, das Rechner, die entsprechende Unterstützung bieten (»ATX«), ausschaltet.
pppd	/usr/sbin	PPP (Point to Point Protokoll) Daemon
pppoectl	/sbin	Anzeige oder Setzen der Parameter eines PPPoE-Devices
pppstats	/usr/sbin	Ausgabe der PPP-Statistik
pr	/usr/bin	Datei-Druckprogramm
praliases	/usr/sbin	Anzeige der System-Mail-Aliases
prefix	/usr/sbin	Konfiguration der Netzwerkinterface-Präfixe
printenv	/usr/bin	Ausgabe der Programmumgebung
printf	/usr/bin	Formatierte Ausgabe
procctl	/usr/sbin	Löschen von procfs-Ereignis-Flags
protoize	/usr/bin	Hilfsprogramm zum C++-Compiler für SYSCALLs
ps	/bin	Anzeige des Prozeß-Status
pstat	/usr/sbin	Anzeige der Datenstrukturen des Systems. Siehe auch *swapinfo*

purgestat	/usr/bin	Link auf */usr/sbin/mailwrapper*
pvcsif	/usr/sbin	Konfiguration des ATM-PVC-Subsystems
pvctxctl	/usr/sbin	Anzeigen oder Ändern der ATM-PVC-Transmitter-Parameter
pwd	/bin	Anzeigen des aktuellen Verzeichnisnamens
pwd_mkdb	/usr/sbin	Erzeuge die Paßwort-Datenbank
quot	/usr/sbin	Anzeige des Platzes auf dem Datenträger, der von jedem Benutzer gebraucht wird
quota	/usr/bin	Anzeige der Festplattennutzung und Begrenzungen
quotacheck	/usr/sbin	Konsistenz von Dateisystem-Quotas prüfen
quotaoff	/usr/sbin	Ausschalten der Dateisystem-Quotas
quotaon	/usr/sbin	Anschalten der Dateisystem-Quotas
racoon	/usr/sbin	IKE- (ISAKMP/Oakley) Schlüsselmanagement-Daemon
radioctl	/usr/bin	Kontrollprogramm für Radio-Tuner
raidctl	/sbin	RAID-Kontrollprogramm
ranlib	/usr/bin	Generieren eines Indexes zu einem Archiv
rarpd	/usr/sbin	Umgekehrter ARP-Daemon
rbootd	/usr/sbin	HP-Remote-Boot-Server
rcmd	/bin	Backend-Treiber für R-Commands
rcorder	/sbin	Startskripten-Analyseprogramm (RC-Reihenfolge)
rcp	/bin	Remote Dateien kopieren (Teil der Berkeley r-Tools)
rcs	/usr/bin	Ändern von RCS-Dateiattributen
rcs2log	/usr/bin	RCS-Hilfsprogramm für den C++-Compiler
rcsclean	/usr/bin	Aufräumen von RCS-Arbeitsdateien
rcsdiff	/usr/bin	Vergleich vwon RCS-Revisionen
rcsfreeze	/usr/bin	Einfrieren der Konfiguration von mit RCS eingecheckten Quellen
rcslog	/usr/bin	Synonym für *rlog*
rcsmerge	/usr/bin	Zusammenführen von RCS-Revisionen
rdate	/usr/sbin	Setzen des Systemdatums von einem Remote-Host
rdist	/usr/bin	Remote-Dateiverteilungsprogramm (Teil der Berkeley r-Tools)
rdump	/sbin	Dateisystem-Backup-Programm (siehe auch *dump*)
rdump_ls	/sbin	Dateisystem-Backup-Programm für LFS-Dateisysteme
readelf	/usr/bin	Analyseprogramm für ELF-Binärdateien
reboot	/sbin	Neustarten des Systems (siehe *fastreboot*, *halt* und *shutdown*)

refer	/usr/bin	Vorbereitung bibliographischer Referenzen für *groff*
renice	/usr/bin	Ändern der Priorität laufender Prozesse
repquota	/usr/sbin	Zusammenfassung der Quotas für ein Dateisystem
reset	/usr/bin	Initialisierung und Neuinitialisierung des Terminals
restore	/sbin	Restaurierung von Dateien oder Dateisystemen aus Backups mit *dump*
rev	/usr/bin	Drehe die Zeilen in einer Datei um
revnetgroup	/usr/sbin	Generieren von reversen Netzgruppendaten
rip6query	/usr/sbin	RIPng Debugging-Tool
rlog	/usr/bin	Ausgabe von Log-Meldungen und anderer Information über RCS-Dateien. Auch unter dem Namen *rcslog* verfügbar.
rlogin	/usr/bin	Remote-Login (Teil der Berkeley r-Tools)
rm	/bin	Löschen von Dateien und Verzeichnisbäumen (siehe auch *unlink*)
rmail	/bin	Verarbeitung von Remote-Mails, die per UUCP eingegangen sind
rmd160	/usr/bin	Checksummen anzeigen(siehe *cksum*)
rmdir	/bin	Löschen eines Verzeichnisses
rmt	/usr/sbin	Remote-Magnetband-Protokoll
rndctl	/sbin	In-Kernel-Zufallsgeneratoren-Management-Programm
route	/sbin	Programm zur manuellen Änderung der Routing-Tabellen
route6d	/usr/sbin	RIP6-Routing-Daemon
routed	/sbin	Netzwerk-Router-Daemon
rpc.yppasswdd	/usr/sbin	Server für das Update von NIS-Paßworten
rpc.boot-paramd	/usr/sbin	Boot-Parameter-Server
rpc.lockd	/usr/sbin	NFS-Dateilocking-Daemon
rpc.pcnfsd	/usr/sbin	Server für (PC)NFS-Authentifizierung und Druck
rpc.statd	/usr/sbin	Hoststatus-Monitoring-Daemon
rpcbind	/usr/sbin	Mapper für Universaladressen nach RPC
rpcgen	/usr/bin	Ein RPC-Protokoll-Compiler
rpcinfo	/usr/bin	Melden von RPC-Informationen
rrestore	/sbin	Wiederherstellung von Dateien oder Dateisystemen aus Backups mit *rdump*

rs	/usr/bin	Größe eines Datenarrays ändern
rsh	/usr/bin	Die Remote-Shell (Teil der Berkeley r-Tools)
rtadvd	/usr/sbin	Router-Werbe-Daemon
rtquery	/usr/sbin	Abfragen von Routing-Daemons nach ihren Routing-Tabellen
rtsol	/sbin	Daemon für ICMPv6-Routing
rtsold	/usr/sbin	Router-Daemon
rup	/usr/bin	Remote-Statusanzeige
ruptime	/usr/bin	Anzeige des Hoststatus der lokalen Maschinen
rusers	/usr/bin	Anfrage, wer in den Maschinen des lokalen Netzes eingeloggt ist
rwall	/usr/bin	Senden einer Nachricht an User, die auf einem Host eingeloggt sind
rwho	/usr/bin	Abfrage, wer an den lokalen Maschinen eingeloggt ist. Dafür muß der *rwhod*-Server laufen
rwhod	/usr/sbin	System-Status-Server
sa	/usr/sbin	Ausgabe der System-Account-Statistik
savecore	/sbin	Sichern eines Core-Dumps des Betriebssystems
scp	/usr/bin	Secure Copy. Remote-Dateikopierprogramm
screenblank	/usr/sbin	Screensaver-Daemon für wscons und FBIO-Maschinen
script	/usr/bin	Erzeugen eines Skripts aller Eingaben in einer Terminal-Sitzung
scsictl	/sbin	Programm zur Bearbeitung von SCSI-Geräten und des Busses.
sdiff	/usr/bin	Finden von Unterschieden in zwei Dateien und interaktive Zusammenführung
sed	/usr/bin	Der Unix-Stream-Editor
sendmail	/usr/sbin	Link auf */usr/sbin/mailwrapper*
send-pr	/usr/bin	Senden von Problem-Reports an eine zentrale Support-Site
sesd	/usr/sbin	Monitor für SCSI-Umgebungsservices
setencstat	/usr/sbin	Setzen des Status von SCSI-Umgebungsservices
setkey	/sbin	Manuelles Anpassen der IPsec-SA/SP-Datenbank
setobjstat	/usr/sbin	Setzen des Objektstatus von SCSI-Umgebungsservices
sftp	/usr/bin	Secure-FTP-Programm
sh	/bin	Der Kommando-Interpreter (Shell)
sha1	/usr/bin	Checksummen anzeigen (siehe *cksum*)

shar	/usr/bin	Erzeugen eines Shell-Archivs von Dateien
shlock	/usr/bin	Erzeugen oder Prüfen einer Lock-Datei für Shellskripten
showmount	/usr/bin	Anzeigen von Remote-NFS-Mounts auf dem Host
shutdown	/sbin	Programm zum Herunterfahren und Neustarten des Systems (siehe auch *halt* und *reboot*)
skey	/usr/bin	Reagieren auf eine OTP-Anfrage (OTP= One Time Password authentification system)
skeyaudit	/usr/bin	User warnen, daß der S/Key bald abläuft
skeyinfo	/usr/bin	Anfordern der nächsten S/Key-Aufforderung
skeyinit	/usr/bin	Ändern eines Paßworts oder Hinzufügen eines Users im S/Key-Authentifizierungs-System
slattach	/sbin	Anbinden einer seriellen Verbindung als Netzwerk-Interface
sleep	/bin	Pausieren der Ausführung für eine anzugebende Zeit
sliplogin	/usr/sbin	Anbinden eines seriellen Netzwerk-Interfaces
slogin	/usr/bin	OpenSSH. Secure Shell Client für den Remote-Login
slstats	/usr/sbin	Anzeigen der IP-Statistik der seriellen Verbindung
soelim	/usr/bin	Entfernen von ».sos« aus *nroff*-Input
sort	/usr/bin	Sortieren von Zeilen in Textdateien
split	/usr/bin	Teilen einer Datei in mehrere Stücke
spray	/usr/sbin	Senden vieler Pakete an einen Host
ssh	/usr/bin	OpenSSH. Secure Shell Client (Remote-Login-Programm)
ssh-add	/usr/bin	Fügt RSA- oder DSA-Identitäten für den Authentifizierungsagenten zu
ssh-agent	/usr/bin	Authentifizierungsagent
ssh-keygen	/usr/bin	Generieren von Authentifizierungsschlüsseln
ssh-keyscan	/usr/bin	Öffentliche SSH-Keys von Hosts sammeln
sshd	/usr/sbin	Secure-Shell-Daemon
stat	/usr/bin	Dateistatus anzeigen
stdether	/usr/sbin	Ein YP-Filterprogramm
stdhosts	/usr/sbin	Ein YP-Filterprogramm
string2key	/usr/bin	Paßwort in einen Key ummappen
strings	/usr/bin	Anzeigen der druckbaren Zeichen in Dateien
strip	/usr/bin	Entfernen der Symbole aus Objektdateien
stty	/bin	Setzen der Optionen für ein Terminal-Device-Interface

su	/usr/bin	Substitutieren einer User-Identität
sum	/usr/bin	Anzeige von Datei-Checksummen und Blockanzahlen
sup	/usr/sbin	Softwareaktualisierungsprotokoll
supfilesrv	/usr/sbin	SUP-Serverprozess
supscan	/usr/sbin	SUP-Serverprozess
sushi	/usr/sbin	Das NetBSD-Verwaltungstool (ab NetBSD 1.6)
swapon	/sbin	Angabe zusätzliche Geräte für Paging und Swapping
sync	/bin	Schreiben des Disk-Caches
sysctl	/sbin	Setzen oder Lesen des Kernel-Status
syslogd	/usr/sbin	Daemon für das Loggen von Systemmeldungen
systat	/usr/bin	Anzeigen der System-Statistiken auf einem Bildschirm
tail	/usr/bin	Anzeigen der letzten Teile einer Datei
talk	/usr/bin	Sprechen mit einem anderen User
tar	/usr/bin	Archivierungs und Restaurierungsprogramm; Tape-Archiver
tbl	/usr/bin	Formatierung von troff-Tabellen
tbrconfig	/sbin	Konfigurieren eines »Token Bucket Regulators« für eine Ausgabeschlange
tcopy	/usr/bin	Kopieren und Prüfen von Magnetbändern
tcpdchk	/usr/sbin	Checkprogramm für die TCP-Wrapper-Konfiguration
tcpdmatch	/usr/sbin	TCP-Wrapper-Entscheidungsprogramm
tcpdump	/usr/sbin	Dumpen des Verkehrs im Netz
tee	/usr/bin	PIPE-T-Stück
telnet	/usr/bin	User-Interface für das telnet-Protokoll (telnet-Client)
test	/bin	Test von Bedingungen, siehe auch *[*
texi2dvi	/usr/bin	Texinfo-Dokumente drucken
texindex	/usr/bin	Sortierprogramm für Texinfo-Indexdateien
tfmtodit	/usr/bin	Erzeugen von Fontdateien für die Verwendung mit *groff -Tdvi*
tftp	/usr/bin	Client für das Trivial-File-Transfer-Protokoll (tftp)
time	/usr/bin	Timerprogramm für zeitgesteuertes Ausführen von Programmen
timed	/usr/sbin	Time-Server-Daemon
timedc	/usr/sbin	Kontrollprogramm für den Time-Server-Daemon
tip	/usr/bin	Zu einem Remote-System verbinden
tn3270	/usr/bin	Vollbildschirm-Remote-Login nach IBM VM/CMS

top	/usr/bin	Anzeigen und Aktualisieren der zehn Top-Prozesse der CPU
touch	/usr/bin	Ändern der Zugriffszeit und Neuanlegen einer Datei
tput	/usr/bin	Interface für die Einstellung des Terminals
tr	/usr/bin	Zeichen in einer Textdatei übersetzen und umsetzen
traceroute	/usr/sbin	Anzeigen des Routings von Paketen zu einem Netzwerk-Host
traceroute6	/usr/sbin	Anzeigen des Routings von IPv6-Paketen zu einem Host
troff	/usr/bin	Dokumenten-Formatierprogramm
trpt	/usr/sbin	Protokollverfolgungsprogramm
trsp	/usr/sbin	Protokollverfolgungsprogramm für SPP-Trace-Records
true	/usr/bin	Rückgabe des Werts »true«
tset	/usr/bin	Initialisierung des Terminals (siehe auch *reset*)
tsort	/usr/bin	Topologisches Sortieren eines gerichteten Graphen
ttxflags	/sbin	Gerätespezifische Flags für Terminals setzen
tty	/usr/bin	Rückgabe des Terminal-Namens des Users
tunefs	/sbin	Beschleunigen eines vorhandenen Dateisystems
ul	/usr/bin	Unterstreichen von Zeilen abhängig von den Unterstreichungszeichen des Terminals
umount	/sbin	Ausbinden von Dateisystemen (siehe *mount*)
uname	/usr/bin	Anzeige von Systeminformationen
uncompress	/usr/bin	Expandieren von *compress*-gepackten Dateien (».Z«-Dateien)
unexpand	/usr/bin	Expandieren von Tabs in Leerzeichen (siehe auch *expand*)
unifdef	/usr/bin	Entfernen von *#ifdef*-Statements aus C-Quelltexten
uniq	/usr/bin	Anzeigen oder Herausfiltern wiederholter Zeilen in einer Datei
units	/usr/bin	Konvertierungsprogramm
unlink	/usr/sbin	Programm zum Entfernen eines Verzeichniseintrags
unvis	/usr/bin	Zurückwandeln der visuellen Repräsentation von Daten in das Original (siehe auch *vis*)
uptime	/usr/bin	Anzeige, wie lange das System läuft
usbdevs	/usr/sbin	Anzeigen der im System vorhandenen USB-Geräte
usbhidaction	/usr/bin	Aktionen auf USB-HID-Geräte durchführen
usbhidctl	/usr/bin	USB-HID-Geräte manipulieren
user	/usr/sbin	Zusammenfassendes User-Management-Programm
useradd	/usr/sbin	Neuen User dem System zufügen
userdel	/usr/sbin	User aus dem System entfernen

userinfo	/usr/sbin	Informationen zu einem User anzeigen
usermod	/usr/sbin	Usereintrag verändern
users	/usr/bin	Auflisten der aktuellen User
uuchk	/usr/sbin	Anzeige der Informationen der UUCP-Konfiguration
uucp	/usr/bin	Unix to Unix Copy (Kopieren im Netzwerk)
uudecode	/usr/bin	Dekodieren von Binärdateien (siehe *uuencode*)
uuencode	/usr/bin	Kodieren von Binärdateien (siehe *uudecode*)
uulog	/usr/bin	Anzeige der UUCP-Logeinträge
uuname	/usr/bin	Auflisten bekannter Remote-UUCP-Sites
uupick	/usr/bin	Empfangen von Dateien, die mit *uuto* übertragen werden
uusched	/usr/bin	Der UUCP-Dateitransfer-Daemon
uustat	/usr/bin	UUCP-Statusabfrage und Kontrolle
uuto	/usr/bin	Senden von Dateien an einen User auf einem Remotesystem
uux	/usr/bin	Remote-Befehlsausführung über UUCP
vacation	/usr/bin	Rückgabe der »Bin nicht da«-Meldung an einen Mail-Sender
vgrind	/usr/bin	Formatieren von Programmquelltexten im *troff*-Stil
vi	/usr/bin	Der visuelle Unix-Editor (siehe auch *view*)
view	/usr/bin	Der visuelle Unix-Editor (siehe auch *vi*)
vipw	/usr/sbin	Editorprogramm für die Paßwort-Datei (verwendet für die Editierfunktion den Systemeditor)
vis	/usr/bin	Anzeige nicht-druckbarer Zeichen in einem visuellen Format
vmstat	/usr/bin	Anzeige der Statistik des virtuellen Speichers
vnconfig	/usr/sbin	Konfiguratieren und Einschalten von Pseudo-Disk-Geräten (wie beispielsweise ISO-Images)
w	/usr/bin	Anzeige, wer eingeloggt ist und was er gerade tut
wall	/usr/bin	Schicken einer Meldung an alle User
wc	/usr/bin	Wort-, Zeilen- und Byte-Zähler
wdogctl	/sbin	Kontrollprogramm für den Watchdog-Timer
what	/usr/bin	Anzeigen, welche Versionen von Objektdateien nötig sind, um eine Datei zu konstruieren
whatis	/usr/bin	Suchen in der *whatis*-Datenbank (siehe auch *apropos*)
whereis	/usr/bin	Suchen von ausführbaren Programme
which	/usr/bin	Suchen einer Programmdatei im definierten Pfad des Users
who	/usr/bin	Anzeige, wer eingeloggt ist (siehe auch *logname* und *whoami*)

whoami	/usr/bin	Anzeige, unter welchem Namen man aktuell eingeloggt ist
whois	/usr/bin	Service für Internet-Domain-Namen und Netzwerknummern
wiconfig	/usr/sbin	Konfigurationsprogramm für WaveLAN/IEEE-Geräte
window	/usr/bin	Fensterumgebung für ASCII-Terminals
wire-test	/usr/sbin	Test für Netzwerk-Interfaces und lokale IP-Adressen
write	/usr/bin	Senden einer Meldung an andere User
wsconscfg	/usr/sbin	Konfiguration virtueller Terminals auf einem wscons-Display
wsconsctl	/sbin	Das Konfigurationsprogramm für die wscons-Konsole
wsfontload	/usr/sbin	Laden einer Font-Bitmap in einen wsfont-Pool oder in ein wscons-Display
wsmuxctl	/usr/sbin	wsmuxe konfigurieren
xargs	/usr/bin	Tool zum Konstruieren von Argumentenlisten beim Starten von Programmen
xstr	/usr/bin	Extrahierung von Strings aus C-Programmen
yacc	/usr/bin	Ein LALR-Parser-Generator. »Yet Another Compiler Compiler«.
yes	/usr/bin	Gibt bis zum Abbruch des Programm den Buchstaben »y« oder den auf der Kommandozeile angegebenen String aus
ypbind	/usr/sbin	NIS-Domain-Bindungs-Daemon
ypcat	/usr/bin	Ausgabe der Werte aller Schlüssel in einer YP-Datenbank
ypinit	/usr/sbin	Aufbauen und Installieren der NIS-Datenbank
ypmatch	/usr/bin	Ausgabe der Werte eines oder mehrerer Schlüssel in einer YP-Datenbank
yppasswd	/usr/bin	YP-Synonym für *passwd*
yppoll	/usr/sbin	Abfrage der Version einer YP-Map von einem YP-Server
yppush	/usr/sbin	Updates der NIS-Datenbank erzwingen
ypserv	/usr/sbin	NIS-Datenbank-Server
ypset	/usr/sbin	Mitteilung an *ypbind*, welcher YP-Serverprozeß zu verwenden ist
yptest	/usr/sbin	Aufruf verschiedener YP-Routinen
ypwhich	/usr/bin	Liefert den Hostnamen des YP-Servers des Map-Masters
ypxfr	/usr/sbin	Lesen einer YP-Karte von einem YP-Server
zcat	/usr/bin	Identisch zu *gunzip* -c
zcmp	/usr/bin	Vergleichen komprimierter Dateien
zdiff	/usr/bin	Vergleichen komprimierter Dateien

zdump	/usr/sbin	Zeitzonen-Dumper
zegrep	/usr/bin	Entspricht *grep -ZE*, siehe *grep*
zfgrep	/usr/bin	Entspricht *grep -ZF*, siehe *grep*
zforce	/usr/bin	Erzwinge die Endung ».gz« für alle *gzip*-Archive
zgrep	/usr/bin	Entspricht *grep -Z*, siehe *grep*
zic	/usr/sbin	Zeitzonen-Compiler
zmore	/usr/bin	Anzeigen komprimierter oder unkomprimierter Textdateien auf dem Terminal
znew	/usr/bin	Umkomprimieren von ».Z«-Dateien in ».gz«-Dateien
zzz	/usr/sbin	System in den Schlafmodus versetzen (APM)

Stichwortverzeichnis

$

$CDPATH 537
$DISPLAY 694
$EDITOR 539
$FCEDIT 539
$GLOBIGNORE 540
$HISTFILE 539
$HISTIGNORE 541
$HISTSIZE 539
$HOME 495
$HOME/.profile 551
$IFS 494
$LOCALBASE 265
$LOGNAME 537
$MAIL 495, 537
$MAILCHECK 538
$MAILPATH 537, 539
$PATH 499
$PKG_PATH 258
$PROMPT_COMMAND 541
$PS1 ..... 538, 540, 541
$PS2 ..... 538, 541
$TERM 535
$TMOUT 540
$VISUAL 540
$XAUTHORITY 697
.Xauthority 697
.xinitrc 633
/16-Netzwerk 127
/24-Netzwerk 127
/8-Netzwerk 127
9wm 653

A

Abhängigkeiten in KDE 660
Access Points 421
Active-Flag im Master Boot Record 37
adjustkernel 182, 295, 298, 608
Administrative Gruppen einrichten 607
Adreßauflösung 815
Adreßreichweiten in IPv6 315
ADSL-Verbindungen 415
Advanced Power Management 190
Advisory Locks 436
Afterstep 652
Aktualisierung des Betriebssystems 293
Alias für Netzwerkadapter 300
Aliase ..... 513 ff
aliases.db 387
amiwm 656
Anonymous FTP-Server 353
Anrufbeantworter 412
Anwender entfernen 603
Anwender hinzufügen 598
Anwendungsschicht 309
Anycast-Adressen 313
Aperture-Driver 170
APM-BIOS 815
AppleTalk-Drucker 489
Application-Gateway 333
apsfilter 708, 718
ap-ssl 340
Arbeitsgruppe 445
arch 171
at 591, 592
Audiogeräte 221, 815
Ausgabe umleiten 573
Ausgabepuffers des Kernels 188
Auslagerungsbereich 157
Auslagerungsdatei 39
Ausloggen an der Shell 494
auth 697
autoboot 729
Automount-Daemon 815
Autoritativer Nameserver 323
awk 603

B

Backticks 511
Backup Domain Controller 445
Bash-spezifische Tasten 525
Basis-Konfiguration des Netzwerks 299
Befehle zusammenfassen 497
Befehlswiederholung in der Shell 517
Belegter Speicherplatz 562
Benutzer anlegen 118
Benutzer einer Gruppe hinzufügen 119
Benutzer zeigen 583
Benutzerdateien bei Aktualisierung anpassen 295
Benutzergruppen 120
Benutzergruppen in NT 466
Benutzergruppen-Attribut 568
Benutzerkonto 118
Benutzeroberfläche nach dem Systemstart 124
Benutzerrechte vergeben 121
Benutzerspezifische Shellskripte 550
Berkeley Fast File System 216
Betriebssystem-Informationen 582
bg 582
Big Endian 427
Bildschirm löschen 557
Bildschirmmeldungen ausschalten 581
Binärkompatibilität zu Linux 215
Binär-Packages 253, 265, 286
BIOS 192
BIOS-Bootstrap 144
blackbox 653
Blockcursor einschalten 214
Bootblock aktualisieren 293
Bootdisketten 23
Booten 297
Booten von Festplatte klappt nicht 48
Bootinformation von entferntem Rechner 218
Bootlader 167
Bootmeldungen ansehen 180
Bootreihenfolge im BIOS 25
Bootsektor 34, 144
Bootselektor 37, 166, 167
Bootswitcher 37

Bourne-Again Shell ... 492
Bourne-Shell ... 491
Brace Expansion ... 507
Broadcast-Adresse ... 127, 311, 313
Browsen über mehrere Subnetze ... 465
Browser-Dienste ... 444
BSD-Partitionen ... 147
build.sh ... 775
Build-Skript ... 776
bunzip2 ... 588
Busadapter ... 193
bzip2 ... 586

C

cat ... 572
CCD-Treiber ... 227
CD mounten ... 111
cd ... 561
CD9660 ... 220
CEST ... 117
CET ... 117
Chat-Verbindung herstellen ... 581
Checksummen ... 825
chflags ... 171
chgrp ... 568
chmod ... 121, 569
chown ... 569
chpass ... 596, 789
chroot ... 352, 558, 789
chroot-jails ... 396
chsh ... 115
clear ... 557
Cloning Interface ... 204
CMOS-Uhr ... 364
CODA ... 221
Common Unix Printing System ... 714
Completion ... 525
compress ... 585
config ... 178, 769
configure ... 279
cp ... 576
cpio ... 588
Cross Compile ... 286, 767, 771, 775
C-Shell ... 491
ctwm ... 656
CUPS ... 714, 753
CUPS-Drucker-Spooling ... 717
CUPS-Administration ... 722
cupsd ... 722
CUPS-Programme ... 722
cvs ... 261, 271

D

Data Fork ... 479
date ... 584
Dateidatum ändern ... 579
Datei-Eigentümer-Attribut ändern ... 569
Dateien angelegen ... 579
Dateien durchsuchen ... 575
Dateien entpacken ... 587
Dateien erzeugen ... 287
Dateien komprimieren ... 585
Dateien konvertieren ... 567
Dateien kopieren ... 567, 576
Dateien löschen ... 577
Dateien sichern ... 590
Dateien suchen ... 564
Dateien übertragen ... 580
Dateien verschieben ... 578
Dateinamen erzeugen ... 507
Dateinamenexpandierung ... 503
Dateisicherung ... 588
Dateistatus ... 834
Dateisystem abhängen ... 603
Dateisystem prüfen ... 123
Dateisysteme ... 216
Dateisystemtabelle ... 156
Dateisystemtyp ... 645
Dateityp ermitteln ... 571
Datei-Zugriffsrechte ändern ... 569
Datenaustausch Unix ↔ Windows ... 441
Datenaustausch ... 427
Datenträger anmelden ... 599
Datum anzeigen ... 584
DCF-77 ... 360
dd ... 567
Debugger ... 118
Default-Gateway ... 300, 332
Default-Route ... 332
Default-Router ... 321
Delegation ... 324
Deutsche Tastatur mit Windows-Tasten ... 612
Deutsche Tastaturbelegung in der Bash ... 523
Deutsches KDE ... 661
Devices anlegen ... 162
df ... 561
DHCP ... 781
DHCP-Client ... 319
DIAGNOSTIC ... 188
Dienst aus dem rc-System starten ... 124
diff ... 563, 574
dirs ... 502
Disketten kopieren ... 567
Diskettenimage ... 25
Disklabel ... 41, 147, 151
Disklabel-Editor ... 151
Display umleiten ... 696
Display-Manager ... 668, 682
Distfile ... 254
Distributionsset bauen ... 767
dmesg ... 180, 706
DNS ... 319, 323, 402
DNS-Datenquelle konfigurieren ... 325
Domain Controller einrichten ... 468
Domain Master Browser ... 445
Domain Name Service ... 402
Domänensystem von Windows ... 444
dosboot.com ... 48, 786
Druckdaemon ... 704
Drucken mit Samba ... 464
Drucken unter MacOS ... 486
Drucker im Netz zur Verfügung stellen ... 703
Drucker plattformübergreifend beschreiben ... 727
Druckeranschluß prüfen ... 706
Drucker-Instanzen in CUPS ... 724
Drucker-Klassen in CUPS ... 724
Druckerport prüfen ... 716
Druckerwarteschlange bearbeiten ... 705
DSL ... 401
du ... 562

Durchsuchen nach Zeichenketten....573
Dynamic Host Configuration Protocol....319, 781
Dynamischer Prompt....538

E

e2fsprogs....156
EasyBoot....50
echo....114, 494, 584
Echtheitszertifikat....339
Editor umdefinieren....116
EHCI....224
Eingegebene Befehle anzeigen....517
Einloggen an der Shell....494
Einwahl und Trennung, automatisch....407
Einwahlskript....407
Einzelne Benutzer aussperren....377
El-Torito....26, 786
Emulationsoptionen....215
Enlightenment....654
Erweiterte Partition....34, 145
ESP Ghostscript....719
ESP Print Pro....715
/etc/fstab....156
/etc/hosts....300, 403
/etc/hosts.allow....304
/etc/hosts.deny....304
/etc/ifaliases....300
/etc/mouse....625
/etc/mygate....300
/etc/myname....300
/etc/profile....551
/etc/rc....124
/etc/rc.conf....124, 125, 300
/etc/rc.local....114, 451
/etc/rc.shutdown....733
/etc/rc.subr....124, 730
/etc/resolv.conf....300
/etc/services....306
/etc/sysctl.conf....161
/etc/X11/XF86Config....646
/etc/XF86Config....646
Ethernet....401
Ethernet-Interface....415
Ethernetkarte....205, 303
Ethernetschicht....309
Exitcode....496
Explizite Gerätenummern im Kernel....185
export....495
ext2, ext2fs....149, 219
ext3....156
External Data Representation....428

F

Farbtiefe....621
fc....522
fcntl....436
fdc....231
fdisk....147
fdisk -B....168
Fehlercodes in der Shell....497
Fehlermeldungen....509
Fenstermanager starten....633
Fenstermanager....652
Fern-Prozeduraufrufe....427
Festplatten-Controller....195
Festplattengeometrien....32
Festplattenreinigung....156
Festplattenverwaltung....143
FFS....216
fg....582
File Flags....169, 171
File Flags setzen....171
File Flags zurücksetzen....172
File Locking....436
file....571
File-Deskriptor-Dateisystem....220
find....564
fips....42
Firewall....276, 334
Flatrate....417
flock....436
Floppy....231
flwm....656
Formatierprogramme....154
Framebuffer....170
Framebuffer-Treiber....643
FreeBSD-Pakete umsetzen....280
Freier Laufwerksplatz....561
fsck....123
fstab-Optionenbereich....161
fstat....438
ftp....580
ftp-Account anlegen....354
ftpchroot....349
ftpd....348, 354
ftpd.conf....358
ftpd.pids-CLASS....349
ftp-Server installieren....348
ftp-Server, Konfigurationsdateien....348
FTP-Upload-Funktionalität....357
ftpusers....349
ftpwelcome....349
fully qualified hostname....300
Funkuhr....367
Funkverkehr....420
fvwm1....656
fvwm2....657
fvwm95....657

G

Gateway....202, 408
gdm....668, 684
gdm.conf....687
Gemeinsames CD-Laufwerk....458
Generic 105-key....612
GENERIC....176
GENERIC_LAPTOP....177
GENERIC_TINY....177
Geöffnete Dateien ersetzen....297
Gerätenummer wechseln....185
Gerätetreiber für Ethernet....209
Gerätetreiber für Wireless LAN....211
Ghostscript....705, 717, 718
GID....119, 429
Globale Shellskripte....550
Gnome....664
Gnome-Desktop starten....665
Gnome-kompatibler Fenstermanager....669
Gnome-Minimalsystem....668
GNU configure....279

GNU Debugger ... 118
Grafikkarte konfigurieren ... 613, 618
Grafiktools ... 726
grep ... 575
groff ... 821
groupadd ... 120
Grundkonfiguration ... 45
Gruppe anlegen ... 120
Gruppenzugehörigkeit von Dateien ... 817
gtar ... 591
gunzip ... 587
gzip ... 585

H

Hard-Mounts ... 434
Hardware-Uhr ... 361
Hauptspeicherverwendung ... 780
head ... 572
Headerdateien für C-Compiler ... 178
Hintergrundprozeß, Status ändern ... 533
Hintergrundprozesse ... 533
History-Funktion ... 517
Hochsicherer Modus ... 170, 186
Homeverzeichnis anlegen ... 119
Hostnamenliste ... 300
hosts.equiv ... 703
hosts.lpd ... 703
https ... 339
hwclock ... 364

I

IceWM ... 657, 669
ICMP ECHO_REQUEST-Pakete ... 303
ICMPv6 ... 312
IDE-Geräte ... 198
IF ... 301
ifconfig ... 302, 303, 319, 336, 402, 420
inetd ... 304 ff
inetd.conf ... 306, 348
Inhalt einer Umgebungsvariable ... 495
Inhalt eines Verzeichnisstacks ... 502
init ... 123, 170
init-Datei ... 372
init-Prozeß ... 729
Inputfilter ... 708
INSECURE ... 186
Installations-Bootfloppies bauen ... 770
Interactive-Session-Skript ... 550
Interfaces auflisten ... 402
Internes Netzwerk ... 403
Internet Printing Protocol ... 714
Internet Protocol Network Address Translation 409
Internet Super Server ... 304
Internet-Domain-Name Server ... 828
Interprozeßkommunikation ... 187
Interruptible Mounts ... 434
ion ... 657
IP ... 309
IP-Adresse 127.0.0.1 ... 320
IP-Adressen ... 127, 320
IP-Aliase ... 302
ipfilter ... 335
IP-Filter-Subsystem ... 205
IP-NAT ... 334, 409
IP-Nummer ... 402
IP-Nummern-Zuweisungsliste ... 300
IPSec ... 202, 420
IPv6 ... 202, 312
IPv6-Adreßtypen ... 312
IPv6-untaugliche Dienste ... 318
ISDN ... 401, 411
isdnctl ... 412
isdnd ... 412
ISDN-Karten ... 411
ISDN-Konfigurationsdateien ... 412
isdnmonitor ... 412
ISDN-Verbindung automatisch trennen ... 414
ISO-9660-Dateisystem einbinden ... 220
ISO-Images einbinden ... 601

J

Job Control ... 581
jobs ... 533, 582
Jobs der aktuellen Sitzung anzeigen ... 533
Jobverwaltung ... 533
Jokerzeichen in der Shell ... 503
Joliet-CDs unter MacOS ... 51
Joliet-Extensions ... 220

K

KAME-Konsortium ... 312
Kanalnummer ... 509
KDE ... 658
KDE starten ... 661
KDE-Minimalsystem ... 663
kdm ... 692
Kernel bauen ... 177, 607, 769
Kernel crosskompilieren ... 232
Kernel, alten starten ... 180
Kernel-Dateisystem ... 220
Kernel-Debugging ... 187
Kernelkompilation ... 175
Kernelkonfiguration für ISDN ... 411
Kernel-Konfiguration im Kernel-Image ... 186
Kernel-Konfigurationsdatei ... 183
Kernel-Konfigurationdateien bearbeiten ... 769
Kernel-lp-Device ... 716
Kernelquellen ... 176
Kernel-Sicherheitsebenen ... 186
Kernel-Variable ... 242 - 251
kernfs ... 157
kill ... 535, 592
Klasse-A-Netz ... 127
Klasse-B-Netz ... 127
Klasse-C-Netz ... 127
known_hosts ... 339
Kommandoketten ... 496
Kommandosubstitution ... 511 ff
Konfigurationsdateien ... 111
Konfigurationsdateien für versch. Rechner ... 176
Konfigurationsdatei generieren ... 183
Konfigurationsdateien aktualisieren ... 295
Konfigurationsdateien der Shell ... 550
Konfigurationsdateien für KDE ... 663
Konfigurationsdateien von X ... 632
Konfigurationsdaten extrahieren ... 192
Konsolen-Treiber ... 212
Korn-Shell ... 492
kpackage ... 267
ksamba ... 443

L

Laptop-Hardware ... 177
Laufwerk einhängen ... 111
ld ... 199
Lese- und Schreibrechte ... 121
LFS-Partition anlegen ... 153
LILO-Bootloader ... 38, 49
Line Printer Daemon ... 703
Link-lokale Adressen in IPv6 ... 314
Links erzeugen ... 578
Linux-Emulation ... 221
List Open Files ... 438
Little Endian ... 427
LKMs nachladen ... 187
lmhosts ... 446
ln ... 578
lo0 ... 303
Load Balancing ... 724
Local Area Network ... 401
Local Master Browser ... 445
localhost ... 301, 403
locate ... 566
lockd ... 436
Login-Skript ... 550
Log-strukturiertes Dateisystem ... 218
Lokaler Loopback-Adapter ... 303
Lokales E-Mail-Postfach herausfinden ... 495
Loopback-Adreßraum ... 127
Loopback-Interface ... 127, 202
lpc ... 705
lpd ... 703
lpq ... 705, 718
lpr ... 705
lprm ... 705, 718
lpt-Treiber ... 200
ls ... 559
lsof ... 438
lwm ... 658

M

m4 ... 386
Mac68k ... 232
Mac-Installer ... 53
Macintosh-Ein-Tasten-Maus ... 66
Macintosh-Partitionsgrößen ... 52
MacOS ... 477
Magic Cookies ... 697
Magicfilter ... 708, 718
Mail Transfer Agent ... 385
make clean-depends ... 265
make deinstall ... 263
make fetch ... 279
make install ... 265
make package ... 279
make patch ... 279
make readme ... 273
make, Umgebungsvariablen ... 771
make-Befehle in Package-Verzeichnis ... 262
MAKEDEV-Skript ... 162
Makefile ... 178
Manpages ... 126
Manual-Pages ... 555
Master Boot Record ... 34, 144
Master-Browser ... 447
MATH_EMULATE ... 178
Mathematischer Coprozessor ... 190
Maus ... 612
Maus-Daemon ... 827
Maustastenzahl ... 616
maxusers ... 185
mbuf-Cluster ... 203
Mehrere Tastaturen oder Mäuse ... 214
mkcert ... 341
mkdir ... 563
Mkfs ... 55
mkhybrid ... 802
mkisofs ... 802
mkxf86config ... 699
mod_ssl ... 339
Modeline ... 636
Modifizierer in der Shell ... 520
modprobe ... 372
Monitorspezifikation in X ... 612
Monolithischer Kernel ... 175
more ... 572
motd ... 349
mount ... 111, 599
mount_mfs ... 780
mount_umap ... 435
mount_union ... 160
mountcritlocal ... 791
mountd ... 430
MSGBUFSIZE ... 188
MTA ... 385
mtree ... 172
Multicast-Adressen ... 313
Multicast-Routing ... 203
Multithreading ... 432
Multi-User-Modus ... 124
mv ... 578

N

Nachladen benutzerdef. Fonts in Konsole ... 214
Nachricht auf Terminal übermitteln ... 580
Namensauflösung in Windows-Netzen ... 446
Namensauflösung von IP-Nummern ... 300
Nameserver ... 403
Nameserver-Client konfigurieren ... 322
NE2000-Netzwerkkarten ... 130
Netatalk ... 203, 477
NetBIOS ... 442
NetBSD-Bootpartition ... 153
NetBSD-Loader ... 786
netstat ... 303, 336
Network Address Translation (NAT) ... 334
Network Information System (NIS) ... 377
Network Time Protocol (NTP) ... 364
Netzmaske ... 127, 319
Netzwerk mit dem Internet verbinden ... 408
Netzwerkadapter abfragen ... 302
Netzwerkadapter ansprechen ... 301
Netzwerkadapter konfigurieren ... 336
Netzwerkadapter-Einstellungen ... 300
Netzwerkdienste beim Systemstart ... 300
Netzwerkdruck mit CUPS ... 725
Netzwerkdrucker ... 449, 711
Netzwerkdruckerstatus ... 489
Netzwerkkarte ... 129
Netzwerk-Konfigurationsdateien ... 300

Netzwerklaufwerke freigeben ... 449
Netzwerkmaske ... 128, 311
Netzwerkprogramme ... 303
Netzwerkprotokolle ... 202
Netzwerkstatus ... 303
Netzwerkverbindungen (aktive anzeigen) ... 303
newfs_mfs ... 780
NFS ... 427, 441
NFS, General-Account beim Mounten ... 429
NFS, Zugriff auf Subnetz beschränken ... 431
NFS, Zugriffsrechte ... 429
NFS-Client einrichten ... 432
NFS-Client einbinden ... 216
nfsd ... 125, 430, 432, 440
nfsiod ... 434
NFS-Mounts ... 433
NFS-Server einrichten ... 430
NFS-Server für Windows ... 441
NFS-Serverteil einbinden ... 218
NFS-Tuning ... 439
NFS-Zugriff durch Administrator ... 429
NIS ... 377
nmbd ... 450
noauto-Einbindung ... 163
nobody ... 431
noclobber ... 509
nodead ... 114
nodump ... 171
nologin ... 349
Non-Interactive-Skript ... 550
nqnfs ... 427
nsswitch.conf ... 403
NT-Domain ... 444
NTFS ... 161
NTFS-Unterstützung ... 220
NTP ... 359
ntpd ... 375
Null-Dateisystem ... 220
obj-Verzeichnisse ... 772

O

OHCI ... 224
Ohne Passwort auf ftp-Server einloggen ... 351
olvwm ... 658
Online-Manual-Pages ... 555
opaque ... 171
OS/2-Bootmanager ... 49
OS-Level im Windows-Netz ... 448
Outputfilter ... 708

P

Package Source aktualisieren ... 271
Package-Infosystem ... 259
Package-Kollektion ... 131
Packages deinstallieren ... 261
Packages erneuern ... 261
Packages installieren ... 256
Packages löschen ... 259
Packages, Info-Zentrum ... 273
Package-Sammlung ... 253
Package-Sourcetree ... 261, 279, 288
Paketdatenbank ... 254
Pakete aus dem Internet installieren ... 258
Pakete offline aktualisieren ... 272
Pakete umsetzen von Free- nach NetBSD ... 280
Pakete von CD-ROM installieren ... 258
Paket-Hierarchie ... 260
Paketverwaltung mit KDE ... 267
Parallele Schnittstellen ... 200
Partition ... 144, 154
Partition nachträglich verkleinern ... 42
Partition Slot ... 147
Partitionen im Disklabel ... 151
Partitionsgröße berechnen ... 31
Partitions-ID ... 34, 36, 149
Partitionstabelle ... 123, 147
Partitionstypen ... 153
passwd ... 115, 555
Passwort, ... 115, 555
pccons ... 212
pcfdate ... 372
PCIBIOS ... 192
PCI-Gerätedefinitionen ... 193
pciide ... 198
Permanent unsicherer Modus ... 169, 186
Permanente Verbindungen ... 417
Pfadnamen anzeigen ... 564
PHY-Treiber ... 212
ping ... 303, 336
Pipelines ... 510
pkg_add ... 131, 253, 256
pkg_delete ... 253, 259
pkg_info ... 253, 259
pkglocate ... 275
Plattenplatze auf FFS beschränken ... 216
Plattformspezifische Kerneloptionen ... 189
Plattformunabhängige Kerneloptionen ... 185
pmlpr ... 712
Point-to-Point-Protokoll ... 303
Polling-Modus ... 200
popd ... 502
Port ... 254
port2pkg ... 281
postinstall ... 295
Postscript-Befehle umwandeln ... 718
Postscript-Interpreter ... 716
poweroff ... 602, 830
PPD-Dateien ... 727
PPP ... 204
ppp ... 401, 402
ppp0 ... 303
pppd ... 404
ppp-Daemon ... 404
ppp-Konfigurationsdateien ... 403
pppoe ... 415
pppoe-Device ... 204
ppp-Verbindung verschlüsseln ... 422
Primäre Partitionen ... 34
Primärprompt ... 538
Primary Domain Controller ... 445
printcap ... 705
Printer Sharing ... 725
proc-Dateisystem, procfs ... 157, 220
Programm zu bestimmten Zeitpunkt starten ... 591
Proxy ... 410
Prozeß beenden ... 535
Prozeß in Hintergrund schicken ... 535
Prozeß terminieren ... 592
Prozeßstatus ... 593, 594
Prozeßverfolgung ... 187

ps....593
Pseudo-Devices....227
Pseudoterminal-Treiber....229
pushd....501
pwd....564

Q

Quelltext des Betriebssystems....767
Quelltextverzeichnis....767
Quota....163, 216, 831
Quoting....500

R

RAID-0....227
RAID-Controller....199
RAIDframe....228
RAM-Disk....157, 218
rawrite....26
rcorder....124, 732
RCS-Revisionen....817
rc-Startsystem....111, 123 ff, 730
Read ahead caching....434
Rechner in ein lokales Netz einbinden....303
Reihenfolge der Skriptabarbeitung....124
Remote Procedure Calls....427
resolv.conf....322, 403
Resolver....323
Resource Fork....479
Reverse Resolving....325
rm....577
rmdir....564
RockRidge-Dateisystem einbinden....220
root....113
root darf sich nicht direkt einloggen....376
root werden....120
root-Passwort vergessen....122
Rootverzeichnis setzen....558
route....303, 336
Router....321, 331
Routing-Tabelle....303
RPC....427, 832
rpc.lockd....437
rpcbind....428
rpcbind-Daemon....379
rpcinfo....437
r-Tools....375
Rückwärtsauflösung....325

S

S/Key....834
Samba....441
Samba starten....451
Samba: zentrale Benutzerverwaltung....466
Samba-Benutzerverwaltung....461
Samba-Clients....473
Samba-Security-Level....461
Samba-Server als Domain Master....465
Sandkasten....352
sappnd....171
sawfish....654, 667
schg....171
Schlafmodus....839
Schnittstelle der Maus....625
Schreibgeschützter Rootbereich....779
Schreibschutz auf Dateiebene....169
Scopes in IPv6....314
Screensaver-Daemon....833
SCSI-Hostadapter....195
Secure Copy....338
Secure Shell....338, 699
Secure-HTTP....740
Securelevel....169, 170
Securelevel 0....170
Securelevel -1....169
Securelevel 1....170
Securelevel 2....170
Securelevel erhöhen....170
Securelevel erniedrigen....170
Sektionsnamen in XF86Config....649
Sekundärprompt....538
Sendmail....385
Sendmail automatisch als Daemon starten....393
sendmail.cf....386
Sendmail virtusertable....388
Serielle Schnittstellen....199
Serielle Schnittstellen am USB....227
Server Message Block....441
Serverdienste....306
Server-Rollen in einem Windows-Netz....445
Session-Skript....550
set....543
setenv....495
Sharity Light....473
Shell wechseln....115
Shellinstanz....514
Shelloptionen der Bash....545
Shelloptionen der Bourne-Shell....543
Shelloptionen der C-Shell....550
Shelloptionen der Korn-Shell....544
Shell-Sitzung, mehrere Programme starten....533
Shell-Variablen....495
showmount....437
shutdown....602
Sicherer Modus....170, 186
Signale an Prozesse senden....535
Simple-Init-System....123
Single User Mode....119, 123, 124, 297
Site Level Aggregate-Identifier....315
skill....606
Slave Nameserver konfigurieren....330
Slave-Server....379
Slice....36, 147
Slice-ID....147
SLIP....204
SMB....441
smbclient....456, 712
smbd....450
smbd.conf....452
smbmount....476
smbpasswd....452
smbprint-Druckfilter....712
SMTP-Relay....396
snice....607
Softdep-Support....216
Soft-Mounts....434
Software nachinstallieren....131
Software-Installation aus Binärarchiven....253
Software-Installation aus Quellen....253
Software-Verwaltung mit sushi....134

sort ... 557
Sortieren ... 557
source routed Datenpakete ... 203
Source-Tree ... 261, 767
Speichergröße setzen ... 190
Spoolbereich ... 703
Spool-Verzeichnis ... 707
Spracherweiterungen ... 283
squid ... 410
ssh ... 337, 375, 422, 694
ssh-Tools ... 376
SSH-Verbindung ... 423
SSLeay-Modul für Perl ... 740
Stack ... 502
Stack rotieren ... 502
Staircase-Effekt ... 707
Standardausgabe ... 508
Standardausgabe als Parameter ... 511
Standardausgaben verbinden ... 510
Standardeingabe ... 508
Standard-Filesystem unter NetBSD ... 216
Standardvariablen der Bash ... 540
Standardvariablen der C-Shell ... 542
Standardvariablen der Korn-Shell ... 539
Standardvariablen der Shell ... 536
startkde ... 661
Startskript ... 514, 550
startx ... 627
Statische Netzwerkadressen ... 319
Sticky Bit ... 122
strings ... 573
Striping-Verfahren ... 227
su ... 120
Subdomain ... 445
Subnetz ... 311
Subshell ... 497
Suchmuster ... 505
sudo ... 607
Super Kill ... 606
sushi-Verzeichnisse ... 133
Swapbereich ... 158
swapctl ... 158
Swapdatei ... 39, 158
swapon ... 158
SWAT ... 443, 471
sysctl ... 241
sysinst sichern ... 47
sysinst ... 28, 43, 177, 293
systat ... 303, 336
System beenden ... 602
system.twmrc ... 670
Systemadministrator ... 113
Systembefehle ... 815
Systemdienste ... 300
Systeminformationen ... 836
System-V Interprozeßkommunikation ... 187
Systemweite Meldungen versenden ... 581
Systemzeit ... 117

T

tail ... 572
talk ... 581
tar ... 590
Tastaturbelegung ... 113
Tastaturbindungen ... 678
Tastaturmodell ... 617
Tastaturtreiber ... 679
Tastenbelegung der Shell ändern ... 530
Tastenkombinationen der Shell ... 524
TCP ... 309
TCP/IP v4 ... 308
TCP/IP-Protokollstack ... 202
tcpdump ... 303, 336
tcupdate ... 295
tee ... 573
Teile eines Filesystems woanders mounten ... 220
Temporäre Dateien löschen ... 265
Terminal reservieren ... 824
Terminaladapter ... 411
testparm ... 455
Text auf Bildschirm ausgeben ... 494
Textdatei ausgeben ... 572
Textdatei erstellen ... 574
Textdateien vergleichen ... 574
Tilde ... 114
Tilde-Ausdruck auflösen ... 506
Tilde-Expandierung ... 505
Time-Server-Daemon ... 835
top ... 594
Tottasten ... 114
touch ... 287, 579
traceroute ... 303, 336
Trusted Domains ... 444
tset ... 836
Tunnelprotokolle ... 204
twm ... 631, 665, 669
twm-Funktionen ... 677

U

uappnd ... 171
Übergangsmechanismen zu IPv6 ... 315
Überlange Kommandos ... 512
Überwachungsprogramme für Netzwerk ... 303
uchg ... 171
ucom ... 226
UCONSOLE ... 186
UHCI ... 224
UID ... 429
umask ... 571
Umgebungsvariablen ... 495
Umlenkung ... 508
umount ... 603
uname ... 582
uncompress ... 587
Unicast-Adressen ... 313
union-Dateisystem ... 221
Unsicherer Modus ... 170, 186
UPN-Rechner ... 818
uptime ... 836
USB ... 224
USB-Ethernet-Adapter ... 226
USB-Geräte ... 836
USB-Maus ... 615
User für Samba anlegen ... 452
useradd ... 119, 598
Userdaten ändern ... 596
userdel ... 603
Usereintrag verändern ... 837
User-Mounts ... 161
users ... 119

UTC 117, 361
utmp 349

V

Variablen in der Shell 495
Verbindung drahtloses/drahtgeb. Netz 421
Verbindung mehrerer Platten zu einer 227
Verbindungen auf- und abbauen 407
Verbindungsskript 404
Verschlüsseltes ftp 338
Verschlüsseltes Passwort 356
Verteiltes Dateisystem 221
Vervollständigung in der Shell 525, 528
Verzeichnis anzeigen 559
Verzeichnis erstellen 563
Verzeichnis löschen 564
Verzeichnis wechseln 114, 561
Verzeichnisaufbau von X 611
Verzeichnisbaum 111
Verzeichnisexport 437
Verzeichnisse vergleichen 563
Verzeichnisstack 501
vi 111, 605
Virtual Private Network 422
Virtuelle Auflösung 613
VPN 202, 422

W

wait 582
wall 581
Warez Kids 357
WaveLAN 420
Webalizer 743
Web-Interface 725
Webmin 739
Webmin installieren 741
Webmin, Apache 761
Webmin, Benutzerverwaltung 749
Webmin, Cronjobs 751
Webmin, Druckerverwaltung 753
Webmin, Internetdienste 756
Webmin, Java-Applets 747
Webmin, Sendmail 759
Webmin, Serverindex 744
Webmin, Softwareverwaltung 751
Webmin, WU-FTPD 764
Webmin, Zugriff kontrollieren 743
Webmin-Module 743
wheel (Gruppe) 120
whereis 565
which 566
who, whoami, who am i 583
wiconfig 420
Wide Area Network 401
Wildcards in der Shell 503
WindowMaker 655
Windows als Druckserver 712
Windows-Partitionen 147
Windows-PC druckt auf NetBSD-Rechner 711
WINS-System 446
Wired down-Konfiguration 185
Workgroup 447
Worttrenner 494
write 580
wscons 212, 630
wsmouse 630
wsmuxe konfigurieren 838
wtmp 349

X

X Printing Panel 726
X Window 609
X Window auf Notebook 643
X Window starten 627
xargs 512
xauth 694, 697
X-Basispakete 610
xcolorsel 679
xdm 682
xdm starten 733
XDMCP 683, 691
XDR 427
xf86cfg 611, 614, 633
xf86config 611
XF86Config 627
XF86Setup 614
xfce 475, 658
XFree86 -configure 699
XFree86 609
xfsamba 475
xhost 694, 696
xinitrc 632
xkbcomp 798
xkeycaps 680
X-Konfigurationsdateien 610
xmodmap 676, 680
X-Server der Version 3.3.6 643
X-Server-Schriften 651
xset 644
X-Sets 609, 774
X-Version 4 läuft nicht 643
xvidtune 622, 638

Y

Yellow Pages 377
YP 377
ypbind-Prozeß 379
YP-Domain einrichten 380
YP-Domain-Namen setzen 379
ypserv 379
YP-System, Befehle 378

Z

Z Shell 492
Zeichenfolge ausgeben 584
Zeileneditor der Bash 521
Zeit dem Kernel bekannt machen 117
Zeitsynchronisation 362
Zeitzone einrichten 363
Zeitzone 45, 186
Zeitzonendatei 117
Zonen-Dateien anlegen 325
Zugang per ssh 377
Zugang zu Rechnerdiensten 304
Zugriff auf unveränderliche Dateien 186
Zugriffseinschränkungen auf Dateien 171
Zugriffsrechte bei NFS 429
Zuletzt eingegebene Zeile wiederholen 518
zzz 839

C&L Computer & Literatur

Herausgegeben von der Redaktion freeX

NetBSD 1.6

Installieren • Konfigurieren • Administrieren

NetBSD ist ein offenes, ausgereiftes und auf vielen Plattformen lauffähiges Unix-Betriebssystem. Es gewinnt stetig an Beliebtheit und besticht durch seinen flexiblen Kernel und seine klare Struktur.

Dieses Buch wurde von Insidern geschrieben, darunter Mitgliedern der NetBSD Foundation. Es erklärt das Betriebssystem und zeigt dem Neuling und Umsteiger, wie es richtig konfiguriert wird. Der erfahrene NetBSD-Anwender profitiert von den Ausführungen zur Systemoptimierung und Sicherheit.

Begonnen wird mit den Grundlagen: der beispielhaften Installation auf mehreren Plattformen, darunter Intel-PC und SparcStation. Anhand dieser Beschreibungen ist auch die Installation auf anderen Plattformen nachvollziehbar. Anschließend wird die Konfiguration des Systems erklärt, das Einrichten des Netzes, der User und verschiedener Daemons sowie des grafischen Systems. Ausführlich wird die Struktur des Kernels und des Paketsystems diskutiert. Damit läßt sich das Betriebssystem optimal an die Hardware und die eigenen Bedürfnisse anpassen.

Besonderer Wert wird in jedem Bereich auf die Netzwerkintegration gelegt – nicht nur die Integration in das lokale Unix- oder Windowsnetz, sondern auch die Einwahl in das Internet via Wählverbindung und DSL wird ausführlich erläutert. Der Abschluß bildet hier die Beschreibung der Einrichtung eines VPN über SSH.

Die CD-ROM ist eine voll funktionsfähige speziell für dieses Buch entwickelte rein CD-basierte Distribution von NetBSD 1.6 für Intel, die keine Installation benötigt. Sie dient dem als Demo, der sich NetBSD vorab ansehen möchte und die Installation auf der Festplatte (noch) scheut. Zugleich kann sie als komfortables Rettungs- und Reparatursystem eingesetzt werden, da sie alle Editoren und sogar die grafische Oberfläche enthält.

Die Themen:

- **Installation auf Intel, Macintosh, SparcStation, Amiga und DecStation**
- **User einrichten, Rechte vergeben, Datenträger verwalten**
- **Das System aktualisieren**
- **Den Kernel optimieren**
- **Das Startsystem konfigurieren**
- **Softwareverwaltung mit dem Package-System**
- **X installieren und anpassen**
- **PPP mit Modem, ISDN und DSL**
- **Netzwerk mit IPv4, IPv6, DNS und Yellow Pages**
- **Datenaustausch mit NFS, FTP, Samba und Netatalk**

Auf der CD

- **NetBSD Live!, CD-basiertes Demo- und Rettungssystem. Mit XFree86 4.2 und deutschem KDE 3, KOffice, Mozilla und Netzwerkanschluß.**

ISBN: 3-936546-00-2

9 783936 546002

EUR 49,90

Coverdesign: HAWA&NÖH